# Yocto Project™: Adding the meta-oracle-java Layer

By Sean D. Liming and John R. Malin
Annabooks

August 2013

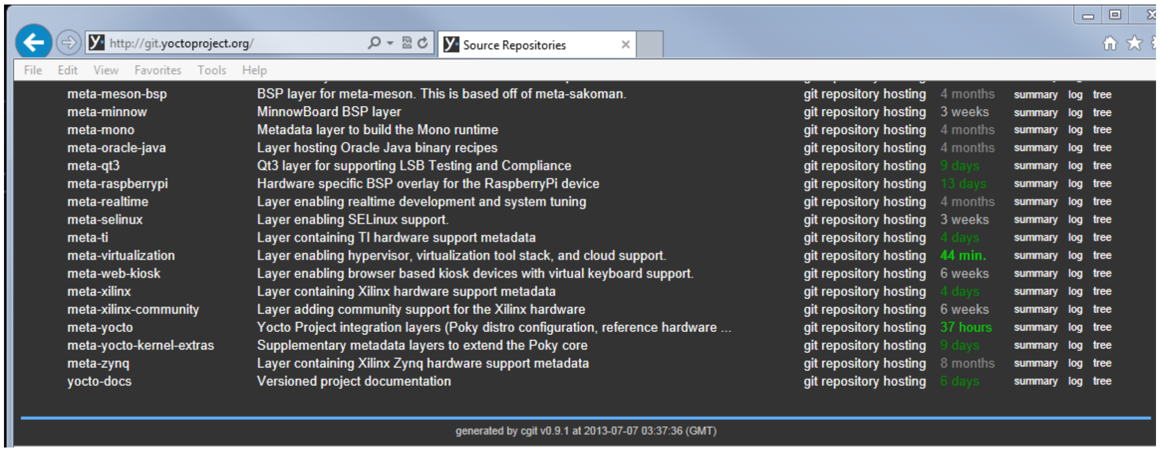
## Java and Linux

Java was designed to be a platform independent managed code engine where applications can be written once and run on many platforms. Best of all it is free for use, which goes great with the royalty-free GNU Linux. Adding Java support to a distribution is commonplace. For the Yocto Project, adding Java means adding a layer to the build to include Java runtime support. A meta-oracle-java layer is available. The normal Yocto Project recipe performs a fetch to upstream sources and downloads the package to be built into the distribution. Automatically fetching the Java packages is a problem since the Java packages require a user to click on an acceptance agreement before the package can be downloaded. The workaround is to download the package and change the recipe to fetch the package locally. This paper discusses a step-by-step solution to update and integrate meta-oracle-java into a Yocto Project build for an x86 processor. This paper was tested with Yocto Project 1.3.x and 1.4.x.

Special thanks goes to the layer developer, Nitin A. Kamble of Intel, who was kind enough to provide feedback.

## Download and Set Up the meta-oracle-java Layer

As of this writing, the meta-oracle-java layer supports Java Standard Edition (SE) 1.7u25. You can see the available layer in the Yocto Project Source Repositories - http://git.yoctoproject.org/

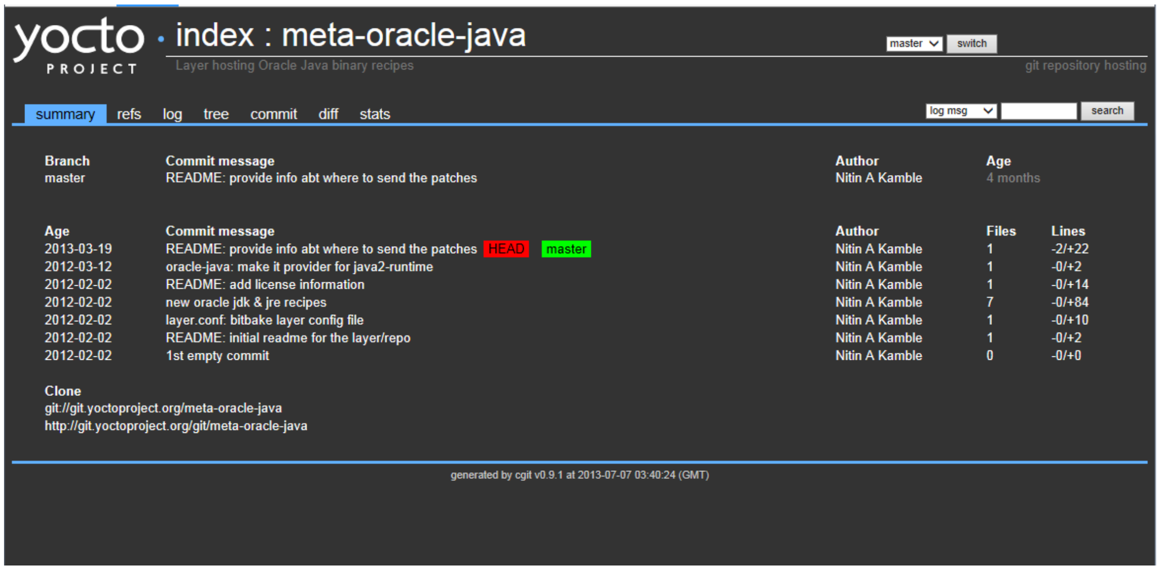


| | | | | | | |
|---|---|---|---|---|---|---|
| meta-meson-bsp | BSP layer for meta-meson. This is based off of meta-sakoman. | git repository hosting | 4 months | summary | log | tree |
| meta-minnow | MinnowBoard BSP layer | git repository hosting | 3 weeks | summary | log | tree |
| meta-mono | Metadata layer to build the Mono runtime | git repository hosting | 4 months | summary | log | tree |
| meta-oracle-java | Layer hosting Oracle Java binary recipes | git repository hosting | 4 months | summary | log | tree |
| meta-qt3 | Qt3 layer for supporting LSB Testing and Compliance | git repository hosting | 9 days | summary | log | tree |
| meta-raspberrypi | Hardware specific BSP overlay for the RaspberryPi device | git repository hosting | 13 days | summary | log | tree |
| meta-realtime | Layer enabling realtime development and system tuning | git repository hosting | 4 months | summary | log | tree |
| meta-selinux | Layer enabling SELinux support. | git repository hosting | 3 weeks | summary | log | tree |
| meta-ti | Layer containing TI hardware support metadata | git repository hosting | 4 days | summary | log | tree |
| meta-virtualization | Layer enabling hypervisor, virtualization tool stack, and cloud support. | git repository hosting | 44 min. | summary | log | tree |
| meta-web-kiosk | Layer enabling browser based kiosk devices with virtual keyboard support. | git repository hosting | 6 weeks | summary | log | tree |
| meta-xilinx | Layer containing Xilinx hardware support metadata | git repository hosting | 4 days | summary | log | tree |
| meta-xilinx-community | Layer adding community support for the Xilinx hardware | git repository hosting | 6 weeks | summary | log | tree |
| meta-yocto | Yocto Project integration layers (Poky distro configuration, reference hardware ... | git repository hosting | 37 hours | summary | log | tree |
| meta-yocto-kernel-extras | Supplementary metadata layers to extend the Poky core | git repository hosting | 9 days | summary | log | tree |
| meta-zynq | Layer containing Xilinx Zynq hardware support metadata | git repository hosting | 8 months | summary | log | tree |
| yocto-docs | Versioned project documentation | git repository hosting | 6 days | summary | log | tree |

generated by cgit v0.9.1 at 2013-07-07 03:37:36 (GMT)

**Figure 1 - Yocto Project Repository**

If you click on meta-oracle-java, the meta-oracle-java branch appears. The summary has a readme file that acknowledges the issues with the user having to manually click to accept an agreement.



index : meta-oracle-java

Layer hosting Oracle Java binary recipes

| Branch | Commit message | Author | Age |
|---|---|---|---|
| master | README: provide info abt where to send the patches | Nitin A Kamble | 4 months |

| Age | Commit message | Author | Files | Lines |
|---|---|---|---|---|
| 2013-03-19 | README: provide info abt where to send the patches HEAD master | Nitin A Kamble | 1 | -2/+22 |
| 2012-03-12 | oracle-java: make it provider for java2-runtime | Nitin A Kamble | 1 | -0/+2 |
| 2012-02-02 | README: add license information | Nitin A Kamble | 1 | -0/+14 |
| 2012-02-02 | new oracle jdk & jre recipes | Nitin A Kamble | 7 | -0/+84 |
| 2012-02-02 | layer.conf: bitbake layer config file | Nitin A Kamble | 1 | -0/+10 |
| 2012-02-02 | README: initial readme for the layer/repo | Nitin A Kamble | 1 | -0/+2 |
| 2012-02-02 | 1st empty commit | Nitin A Kamble | 0 | -0/+0 |

Clone
git://git.yoctoproject.org/meta-oracle-java
http://git.yoctoproject.org/git/meta-oracle-java

generated by cgit v0.9.1 at 2013-07-07 03:40:24 (GMT)

**Figure 2 – meta-oracle-java branch**

1. Make sure you have set up Yocto Project per the quick start guide.
2. Open a terminal window.
3. Change directory to the poky-danny-8.0.x folder (or poky-<release>-<ver>).
4. Run the following git command to download the layer:

   ```
   $git clone git://git.yoctoproject.org/meta-oracle-java
   ```

The git command will download the meta-oracle-java layer into the folder with the rest of the layers. The layer supports four different packages: Java SE JDK i586, Java SE JDK x86-64, Java SE JRE i586, Java SE JRE x86-64. The seven files, whose relationship is diagramed in Figure 3, are the heart of the Java layer. The recipe (.bb) files have dependences on the include (.inc) files. Each file has a specific roll to play.

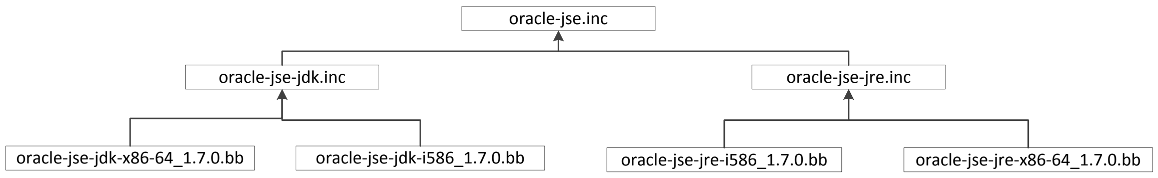

oracle-jse.inc
oracle-jse-jdk.inc
oracle-jse-jre.inc
oracle-jse-jdk-x86-64_1.7.0.bb
oracle-jse-jdk-i586_1.7.0.bb
oracle-jse-jre-i586_1.7.0.bb
oracle-jse-jre-x86-64_1.7.0.bb

**Figure 3 – Recipe Architecture**

The oracle-jse.inc file is used by 4 recipe files to perform the actions of installing the package. The oracle-jse.inc file includes the license file checksums and the do_install(), which performs the actual installation. The do_install creates the java directory and copies the files to a version folder under the /user directory. It then makes a symbolic link to the Java folder.

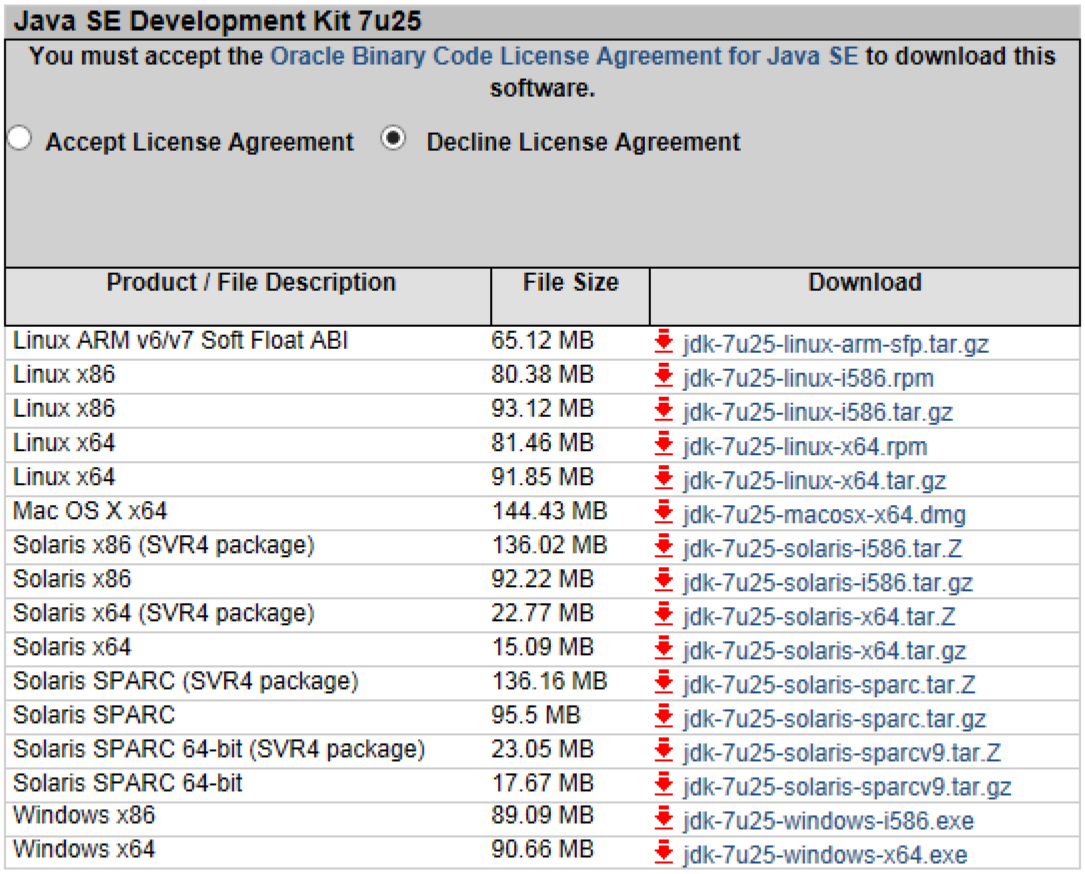
```
:
:
do_install () {
       install -d -m 0755  ${D}/usr/${JDK_JRE}${PV}_${PV_UPDATE}
       cp -a ${S}/${JDK_JRE}${PV}_${PV_UPDATE} ${D}/usr/
       ln -sf ${JDK_JRE}${PV}_${PV_UPDATE}     ${D}/usr/java
}
:
:
```

The various variables come from the different files.

{JDK_JRE} is set by oracle-jse-jdk.inc or oracle-jse-jre.inc.
{PV} comes from the recipe version, which is 1.7.0.
{PV_update} comes from one of the four recipe files to the value of the update being used.

The oracle-jse-jdk.inc and oracle-jse-jre.inc simply add the type of Java being installed: JRE or JDK. The recipes (.bb) control the action to download the packages and set up the update version value. Since a fetch will not work with a manual acceptance of the user agreement, the workaround is to download the packages and put them into the local layer folders. Only download the package that you need.

5. Open an Internet browser.
6. Go to the download site: http://www.oracle.com/technetwork/java/javase/downloads/index.html
7. Locate the package you want to download. For this example, JSE JDK i586 will be used – jdk-7u25-linux-i586.tar.gz.



**Java SE Development Kit 7u25**

You must accept the Oracle Binary Code License Agreement for Java SE to download this software.

Accept License Agreement [✓ Decline License Agreement]

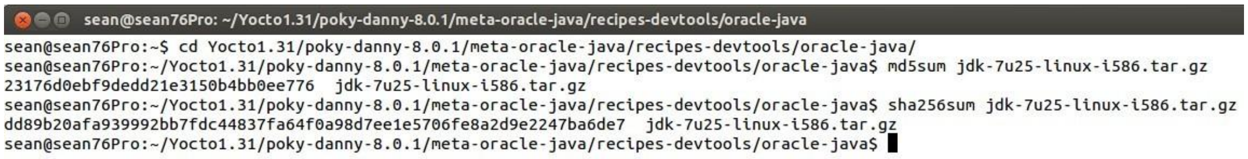
| Product / File Description | File Size | Download |
|---|---|---|
| Linux ARM v6/v7 Soft Float ABI | 65.12 MB | jdk-7u25-linux-arm-sfp.tar.gz |
| Linux x86 | 80.38 MB | jdk-7u25-linux-i586.rpm |
| Linux x86 | 93.12 MB | jdk-7u25-linux-i586.tar.gz |
| Linux x64 | 81.46 MB | jdk-7u25-linux-x64.rpm |
| Linux x64 | 91.85 MB | jdk-7u25-linux-x64.tar.gz |
| Mac OS X x64 | 144.43 MB | jdk-7u25-macosx-x64.dmg |
| Solaris x86 (SVR4 package) | 136.02 MB | jdk-7u25-solaris-i586.tar.Z |
| Solaris x86 | 92.22 MB | jdk-7u25-solaris-i586.tar.gz |
| Solaris x64 (SVR4 package) | 22.77 MB | jdk-7u25-solaris-x64.tar.Z |
| Solaris x64 | 15.09 MB | jdk-7u25-solaris-x64.tar.gz |
| Solaris SPARC (SVR4 package) | 136.16 MB | jdk-7u25-solaris-sparc.tar.Z |
| Solaris SPARC | 95.5 MB | jdk-7u25-solaris-sparc.tar.gz |
| Solaris SPARC 64-bit (SVR4 package) | 23.05 MB | jdk-7u25-solaris-sparcv9.tar.Z |
| Solaris SPARC 64-bit | 17.67 MB | jdk-7u25-solaris-sparcv9.tar.gz |
| Windows x86 | 89.09 MB | jdk-7u25-windows-i586.exe |
| Windows x64 | 90.66 MB | jdk-7u25-windows-x64.exe |

**Figure 4 - Java Downloads**

8. You will need to click on the “Accept License Agreement” radio button before you can download the file.
9. Once the agreement is accepted, click on the .tar.qz file to download.
10. Place the .tar.gz file in the folder with the rest of the recipes.
11. The next step is to modify the recipe file to look for the package in the local folder rather than in the internet resources. We also need to change the update version and set the checksums. Open the oracle-jse-jdk-i586_1.7.0.bb file (or the file related to your package).
12. If the release is later than u25 Change, the PV_UPDATE value to the current release.
13. Comment (#) out the SRC_URI line to prevent a fetch of the resource from the Internet.
14. Add a line to fetch the package from with the layer directory:

```
SRC_URI = "file://${THISDIR}/jdk-7u25-linux-i586.tar.gz"
```

15. If this is a different release than u25, the last step is to update the checksum values for the new zip file. Open a terminal.
16. Change directory to the package.
17. Run md5sum and sha256sum to generate new values.


sean@sean76Pro: ~/Yocto1.31/poky-danny-8.0.1/meta-oracle-java/recipes-devtools/oracle-java

```
sean@sean76Pro:~$ cd Yocto1.31/poky-danny-8.0.1/meta-oracle-java/recipes-devtools/oracle-java/
sean@sean76Pro:~/Yocto1.31/poky-danny-8.0.1/meta-oracle-java/recipes-devtools/oracle-java$ md5sum jdk-7u25-linux-i586.tar.gz
23176d0ebf9dedd21e3150b4bb0ee776  jdk-7u25-linux-i586.tar.gz
sean@sean76Pro:~/Yocto1.31/poky-danny-8.0.1/meta-oracle-java/recipes-devtools/oracle-java$ sha256sum jdk-7u25-linux-i586.tar.gz
dd89b20afa939992bb7fdc44837fa64f0a98d7ee1e5706fe8a2d9e2247ba6de7  jdk-7u25-linux-i586.tar.gz
sean@sean76Pro:~/Yocto1.31/poky-danny-8.0.1/meta-oracle-java/recipes-devtools/oracle-java$
```

**Figure 5 - Create New Checksum Values**

18. Update the checksum values in the recipe (.bb) file.
19. Save the file.

Here is what the oracle-jse-jdk-i586_1.7.0.bb file should look like

```
require oracle-jse-jdk.inc

PR = "r0"
PV_UPDATE = "25"

#SRC_URI = "http://download.oracle.com/otn-pub/java/jdk/7u2-b13/jdk-7u2-
linux-i586.tar.gz"

SRC_URI = "file://${THISDIR}/jdk-7u25-linux-i586.tar.gz"

SRC_URI[md5sum] = "23176d0ebf9dedd21e3150b4bb0ee776"
SRC_URI[sha256sum]                                                    =
"dd89b20afa939992bb7fdc44837fa64f0a98d7ee1e5706fe8a2d9e2247ba6de7"
```

The license checksum in the oracle-jse.inc file will also need updating, but the only known method is to pick up the error during the build.

## Setting Up the Build

Now, we need to set up the build files (bblayers.conf and local.conf) to include the layer and the specific package.

1. Open a terminal and change to the folder with the poky directory.
2. Create the project:

```
$source poky-<release>-<ver>/oe-init-build-env myproject
```

3. Edit the bblayers.conf file to include the path the meta-oracle-java layer. For example:

```
BBLAYERS ?= " \
  /home/sean/Yocto1.31/poky-danny-8.0.1/meta \
  /home/sean/Yocto1.31/poky-danny-8.0.1/meta-yocto \
  /home/sean/Yocto1.31/poky-danny-8.0.1/meta-yocto-bsp \
  /home/sean/Yocto1.31/poky-danny-8.0.1/meta-intel \
  /home/sean/Yocto1.31/poky-danny-8.0.1/meta-intel/meta-cedartrail \
  /home/sean/Yocto1.31/poky-danny-8.0.1/meta-oracle-java \
  "
```

4. Save and close the file.
5. Edit the local.conf file and add the following to the end of the file to include the JSE JDK i586 package:

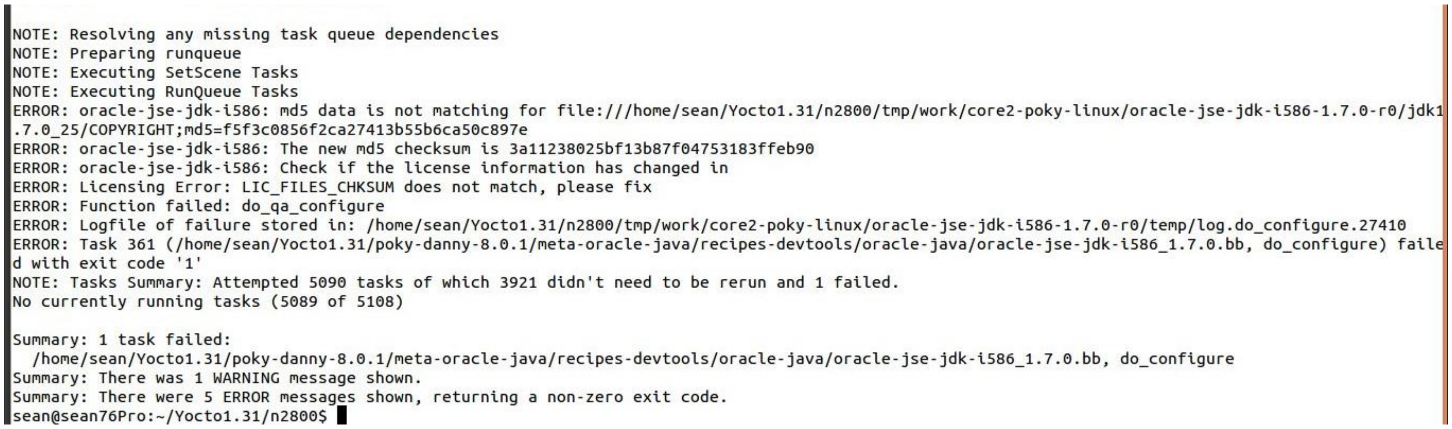
```
LICENSE_FLAGS_WHITELIST += "oracle_java"
```

```
IMAGE_INSTALL_append += " oracle-jse-jdk-i586"
```

6. Save and close the file.
7. Perform a build of an image:

```
$bitbake –k core-image-x11
```

You might run into a license error as shown in Figure 6. The license error may come up only if you are changing the java version of the recipe. The error also shows the solution by providing the correct checksum, which you can copy from the terminal output.



```
NOTE: Resolving any missing task queue dependencies
NOTE: Preparing runqueue
NOTE: Executing SetScene Tasks
NOTE: Executing RunQueue Tasks
ERROR: oracle-jse-jdk-i586: md5 data is not matching for file:///home/sean/Yocto1.31/n2800/tmp/work/core2-poky-linux/oracle-jse-jdk-i586-1.7.0-r0/jdk1
.7.0_25/COPYRIGHT;md5=f5f3c0856f2ca27413b55b6ca50c897e
ERROR: oracle-jse-jdk-i586: The new md5 checksum is 3a11238025bf13b87f04753183ffeb90
ERROR: oracle-jse-jdk-i586: Check if the license information has changed in
ERROR: Licensing Error: LIC_FILES_CHKSUM does not match, please fix
ERROR: Function failed: do_qa_configure
ERROR: Logfile of failure stored in: /home/sean/Yocto1.31/n2800/tmp/work/core2-poky-linux/oracle-jse-jdk-i586-1.7.0-r0/temp/log.do_configure.27410
ERROR: Task 361 (/home/sean/Yocto1.31/poky-danny-8.0.1/meta-oracle-java/recipes-devtools/oracle-java/oracle-jse-jdk-i586_1.7.0.bb, do_configure) faile
d with exit code '1'
NOTE: Tasks Summary: Attempted 5090 tasks of which 3921 didn't need to be rerun and 1 failed.
No currently running tasks (5089 of 5108)

Summary: 1 task failed:
  /home/sean/Yocto1.31/poky-danny-8.0.1/meta-oracle-java/recipes-devtools/oracle-java/oracle-jse-jdk-i586_1.7.0.bb, do_configure
Summary: There was 1 WARNING message shown.
Summary: There were 5 ERROR messages shown, returning a non-zero exit code.
sean@sean76Pro:~/Yocto1.31/n2800$
```

**Figure 6 - License Error**

8. To fix the error, edit the oracle-jse.inc and update the COPYRIGHT;md5 checksum value with the correct version (in this example the original md5 checksum was f5f3c0856f2ca27413b55b6ca50c897e and should be corrected to 3a11238025bf13b87f04753183ffeb90).
9. Save and close the file.
10. Restart the build.

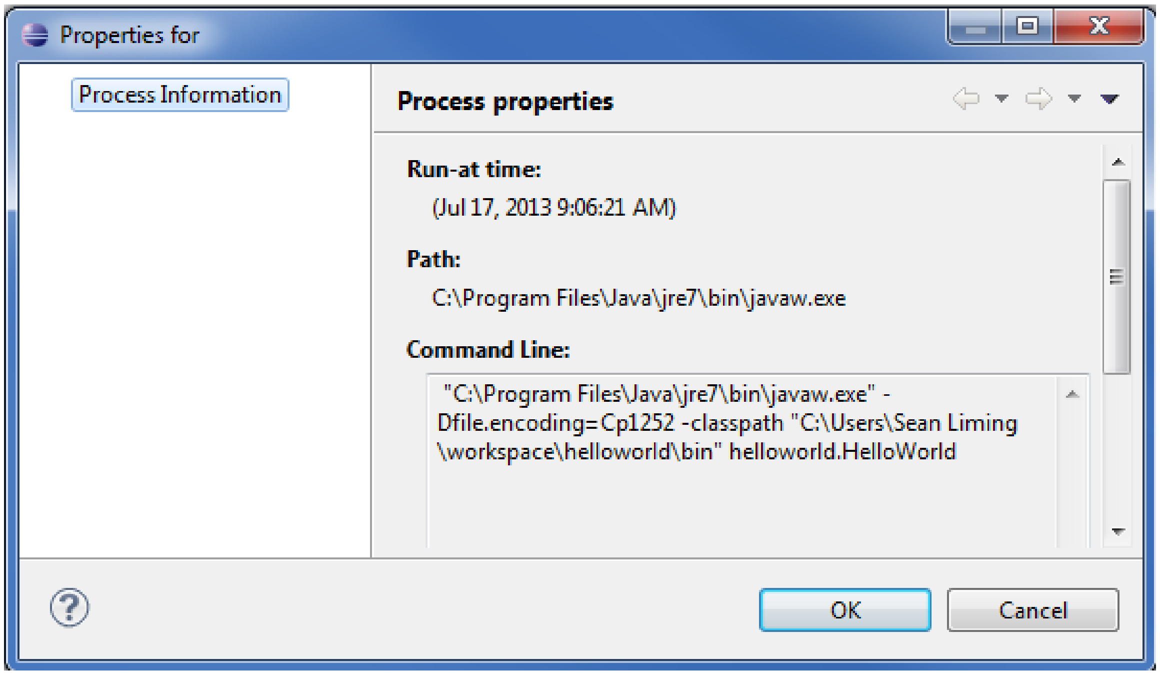
## Final Configuration

Once the distribution has been deployed, the JDK should be installed under the /usr directory. There should be two folders /usr/java and /usr/jdk1.7.0_25. The final configuration step is to create a link to the /usr/bin directory so Java can be executed from anywhere:

1. Open a terminal if one is not open.
2. Enter the following command to install Java in the /usr/bin folder (this step is purposely completed so that it will not clash with open Java recipes):

```
$update-alternatives --install /usr/bin/java java /usr/java/bin/java 1
```

If you write Java applications in Eclipse, starting the application requires using the class name that contains the main() functions as part of the run command. To locate the command line to run the application, first run the application. Then go to the Debug perspective and right-click on the line that says “Terminate, exit value…’ and select properties for the context menu. For example, Figure 7 shows the command line run properties of an application called helloworld. The application was developed on a Windows host. The helloworld application contains a class called HelloWorld that has the main() function.



Run-at time:
(Jul 17, 2013 9:06:21 AM)

Path:
C:\Program Files\Java\jre7\bin\javaw.exe

Command Line:

```
"C:\Program Files\Java\jre7\bin\javaw.exe" -
Dfile.encoding=Cp1252 -classpath "C:\Users\Sean Liming
\workspace\helloworld\bin" helloworld.HelloWorld
```

**Figure 7 - Locate Command Line String**

With some minor adjustments for Linux, the command line for running the task in the Linux image would be as follows:

```
$java -Dfile.encoding=Cp1252 -cp /<my-class-path>/. helloworld.HelloWorld
```

## GUI Application Shell

As a final note, a graphical java application can be launched as the shell interface to the user. Using the core-image-x11 / Yocto Project 1.3.1 distribution as an example, the steps below will launch the application on startup:

1. For this example, a java project called GUITest1 is developed in Eclipse. The project is placed in the distribution folder: /usr/java-apps. To make scripts easier to access, the first step is to create a script to hold the java command line to run the application:

```
#!/bin/sh
java -Dfile.encoding=Cp1252 -cp /usr/java-apps/GUITest1/bin/. guitest1.GUITest1
```

2. Make the script executable:

```
$ chmod +x guijava
```

3. To start the application as the shell, a script needs to be created for X. core-image-x11 has an x-session-manager script that looks for a script called ‘session’ in a /etc/mini_x folder. If the script is not found, a default terminal application is run. Create a folder /etc/mini_x.
4. Create a script called “session” in the folder /etc/mini_x:

```
#!/bin/sh
/usr/bin/guijava&
Exec matchbox-window-manager
```

5. Make the script executable:

```
$ chmod +x session
```

6. Reboot the machine and the graphical java application should launch on startup

## Summary

The meta-oracle-java layer provides a base structure to integrate Java into a distribution. As Java is updated, the latest version can be integrated by making some changes to the recipe files and manually downloading the java package ahead of time.

Feedback welcome: http://www.annabooks.com/Contact.html