Configuring Eclipse for OCAP stack debugging

The following steps describe how to create an Eclipse project for the RI OCAP stack source code that will enable debugging of the OCAP java source code

1. Create a new project

- File->New->Java Project
- Enter Project name, and select "Finish" to accept remaining defaults

2. Import OCAP source

- Right click on newly created project from step 1 & select "Properties"'
- Click "Java Build Path" in the left pane of the properties form
    - Click on the "Source" tab and then select the "Link Source..." Button
        - Navigate to and select $OCAPROOT/ri/RI_Stack/java/src/base
        - use the default folder name
        - Select the "OK" button

... Repeat above for ds, dvr, fp, and hn in directories located in the $OCAPROOT/ri/RI_Stack/java/src/

3. Add Exclusion patterns

- In the "Source" tab of the "Java Build Path", double-click the <Project>/base entry, click Next in the Edit source folder window.
- Next to the Exclusion patterns pane, click Add Multiple...
- Browse to org/cablelabs/impl/dvb/ui, using the arrows next to the folder to descend into the child directory.
- Control-click
    - DVBBufferedImagePeer2.java
    - DBGraphicsImpl.java
    - DBGraphicsImpl2.java

  to highlight them.

- Browse to org/apache/log4j/rule, control-click to highlight "LikeRule.java". That's 4 highlighted exclusions. Click OK, Finish.
- Back at the Source tab, double-click the <Project>/dvr entry, click Next in the Edit source folder window.
- Next to the Exclusion patterns pane, click Add Multiple...
- Browse to org/ocap/shared/media, control-click BeginningOfContentEvent.java and EndOfContentEvent.java, click OK, Finish.

4. Configure the project to work with the OCAP code

- Still in the project Properties window, click "Java Build Path" in the left pane of the properties form then select the "Libraries" tab
    - Remove "JRE System Library" (select and hit "Remove" button)
    - In order to compile OCAP code succesfully add the following external jar files
        - Hit "Add External JARs..." button
        - browse to & select $OCAPROOT/bin/lib/ocap-stub.jar
        - browse to & select $OCAPROOT/tools/generic/cybergarage/cybergarage.jar
        - browse to & select $OCAPROOT/tools/generic/NanoXML-2.2.3/lib/nanoxml.jar
        - browse to & select $OCAPROOT/tools/generic/java/pbp11.jar

5. Configure the compiler

- Click "Java Compiler" in the left pane of the properties form
    - Make sure "Enable project specific settings" is checked
    - Make sure "Use default compliance settings" is unchecked
    - set all compiler compliance levels to 1.4

- Click OK button at the bottom to exit the properties form

6. Add line numbers to output

- In the Eclipse "Window" menu, select Preferences.
- In General->Editors->Text Editors (parent), check "Show line numbers", click OK.

7. Configure the RI Stack to accept remote debugging connections

- Make sure the follow entries are un-commented in $OCAPROOT/bin/$OCAPTC/env/mpeenv.ini
    - VMOPT.19=-Xrunjdwp:transport=dt_socket,address=8000,server=y,suspend=y
    - VMOPT.20=-Xdebug

8. Create a remote JVM debug launcher configuration in eclipse

- Click the down arrow next to the debug icon (small green bug icon, 4th from the left, along the top eclipse icon panel)
- Select "Remote Java Application", and click the "new" icon (upper left hand side of the form ... piece of paper with + sign in upper right corner)
- OK Keep all of the defaults
- Leave this form

9. Launch and debug the code

- cd $PLATFORMROOT; runRI.sh
- as soon as you see "Listening for transport dt_socket at address: 8000" in the output console, hit the "Debug" button on the "Launch Debug Configurations" form