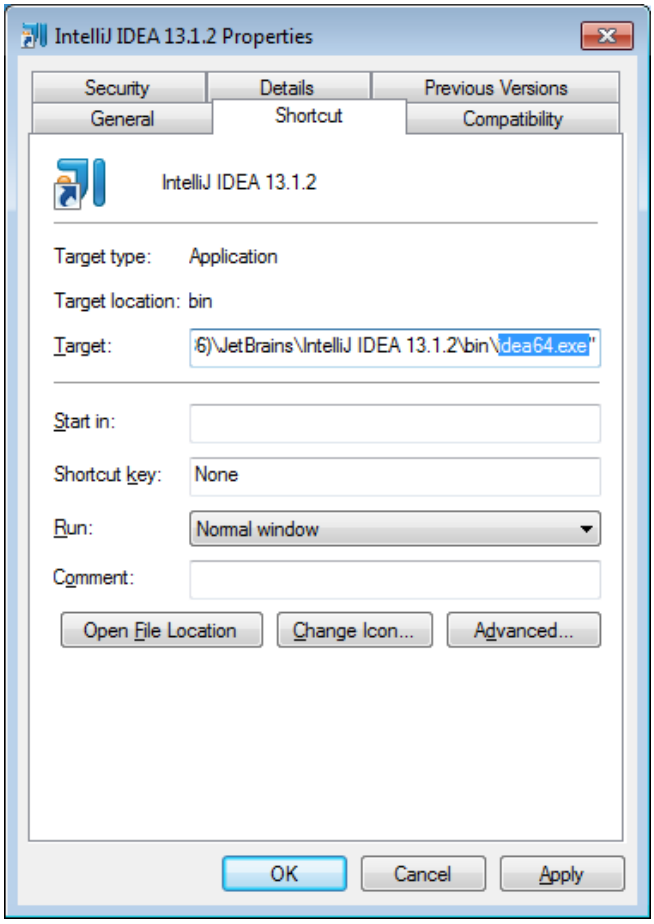
UPM 3 Setup Instructions

Following are the instructions to set up a local developer workstation for UPM 3 development.

1. The software required for UPM3 development is stored on the network drive: \\msp00100\shared\Software\UPM3
   If you don't have access to that drive, submit a SATS request for your Windows ID to be added to the group: UHTDevel

2. **IntelliJ 13:** UPM 3 development is dependent on using IntelliJ. In order to avoid performance issues with IntelliJ, particularly with UPM2 or having multiple projects open, use the 64-bit version and increase the memory settings.

   Change the desktop and toolbar icon to point to idea64.exe:



In the IntelliJ installation /bin directory, edit the idea64.exe.vmoptions file (e.g., C:\Program Files (x86)\JetBrains\IntelliJ IDEA 13.1.2\bin\idea64.exe.vmoptions). Suggested values for the top 4 lines:

-Xms128m
-Xmx1536m
-XX:MaxPermSize=768m
-XX:ReservedCodeCacheSize=128m

3. **Maven:** UPM 3 uses maven for library dependency management and builds.

    a. Download and unzip the officially-supported UHG Maven version (currently 3.0.4) from the shared drive: http://uniteddocs.uhc.com/itservices2/AETeam/UPM/Shared%20Documents/UPM3/Downloads/apache-maven-3.0.4.zip

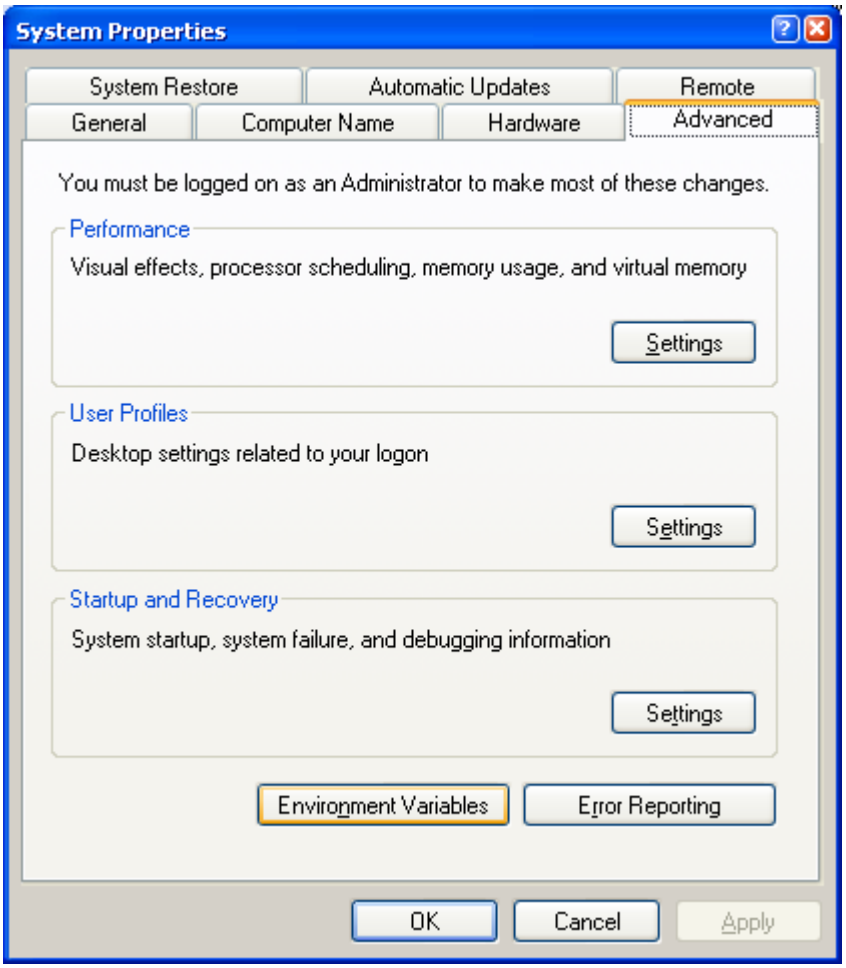
    b. Log in to Artifactory to get your encrypted password:

        i. Go to http://repo1.uhc.com/artifactory/webapp/login.html
        ii. Log in using your Windows login
        iii. Click on your username in the upper right corner
        iv. Type in your Current Password and click Unlock
        v. This will give you what you should put in the “server” section of the maven settings.xml file. Replace username/password in your local settings.xml file (NOTE—only change the username/password tags, leave the id as “artifactory” even though it gives you something else to paste that has ${server-id}—this will not work):

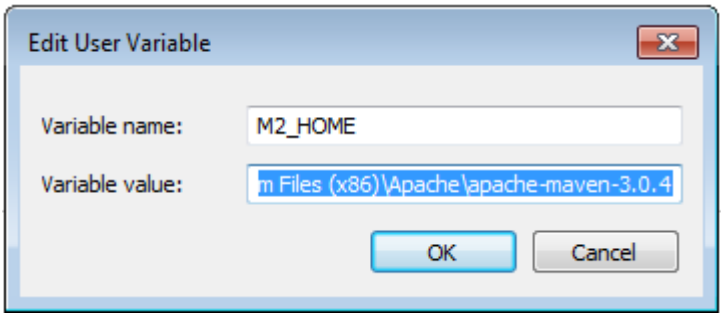
```
<server>
    <id>artifactory</id>
    <username>[userId]</username>
    <password>[encrypted.pwd.from.artifactory]</password>
</server>
```

        vi. You will need to do this every time you change your Windows password.

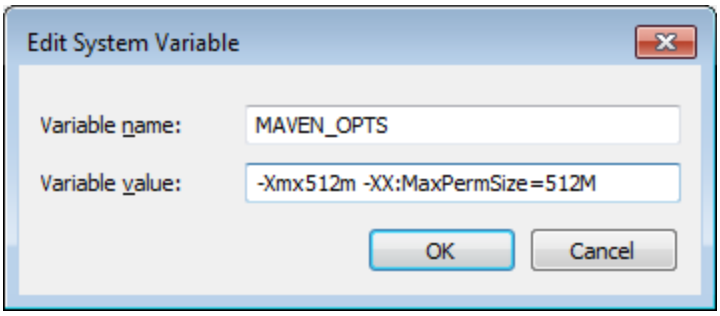
    c. Set up the system variables for Maven. In the System Control Panel, select the “Advanced” tab, then “Environment Variables”:



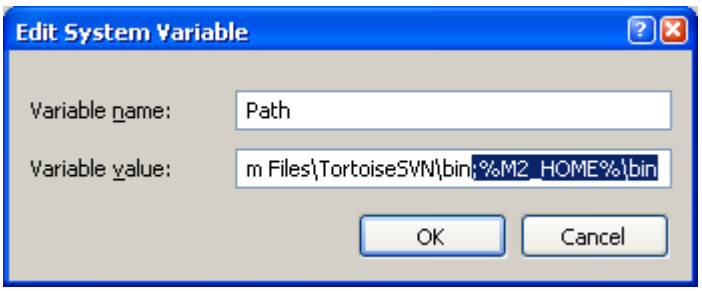
d. Add a new System variable “M2_HOME” specifying the maven directory:



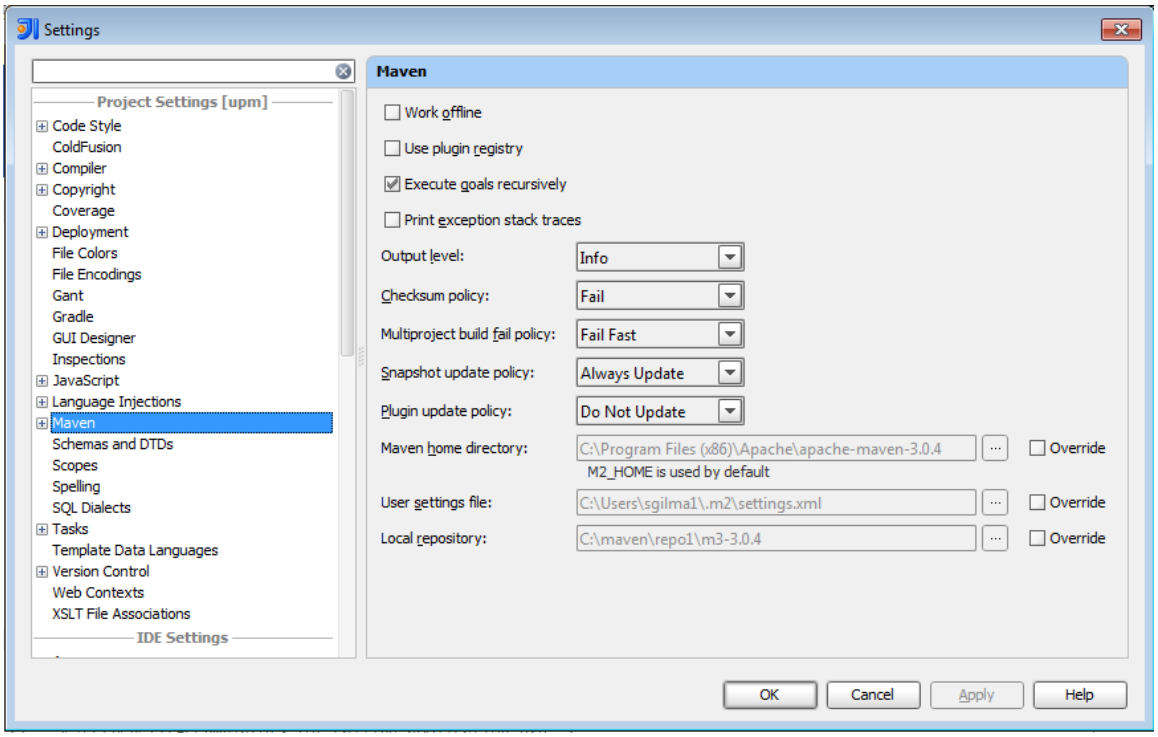
e. Add a new System variable “MAVEN_OPTS” set to “-Xmx512m -XX:MaxPermSize=512M” to increase the default memory setting:



f. Append ";%M2_HOME%\bin" to the end of the "Path" system variable:



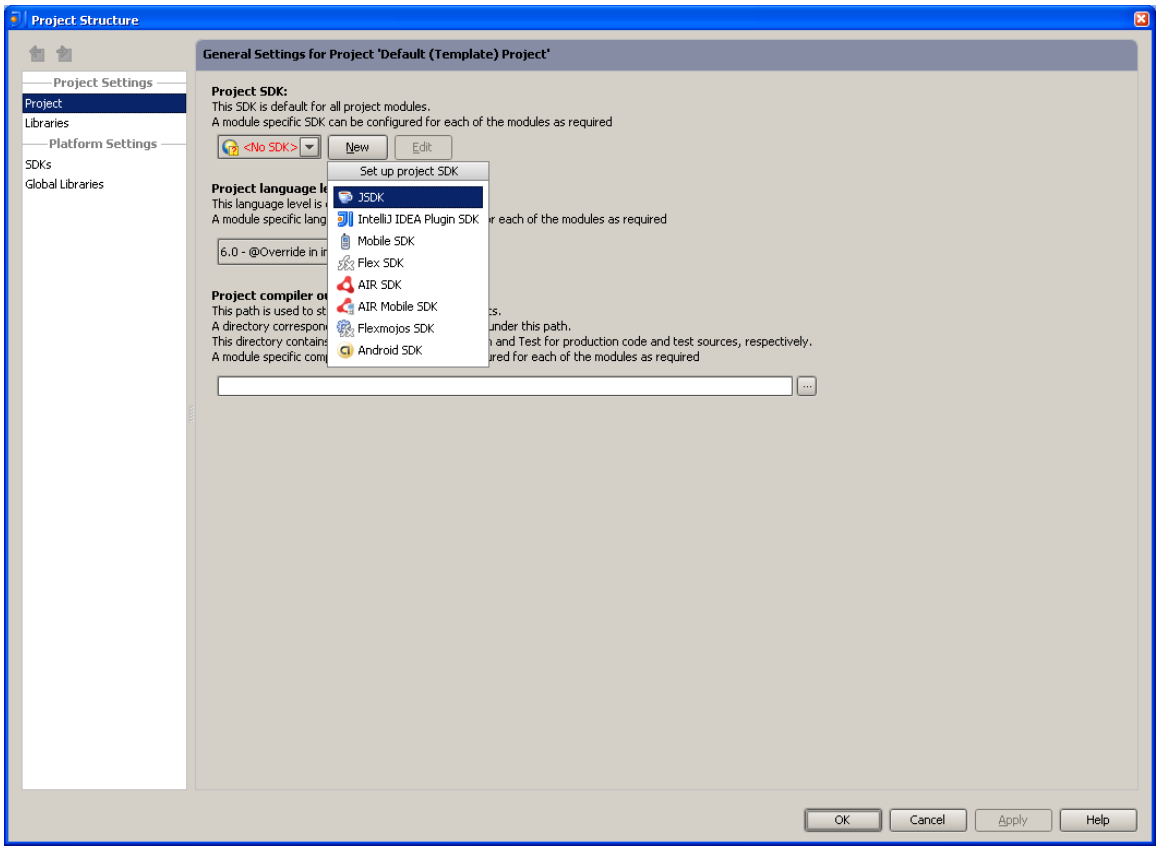
g. If you had IntelliJ started already, shut it down and restart so it picks up the M2_HOME system variable that it uses for the maven installation. Verify this after restarting by looking in Settings under Maven (note that the Maven home directory is filled in):



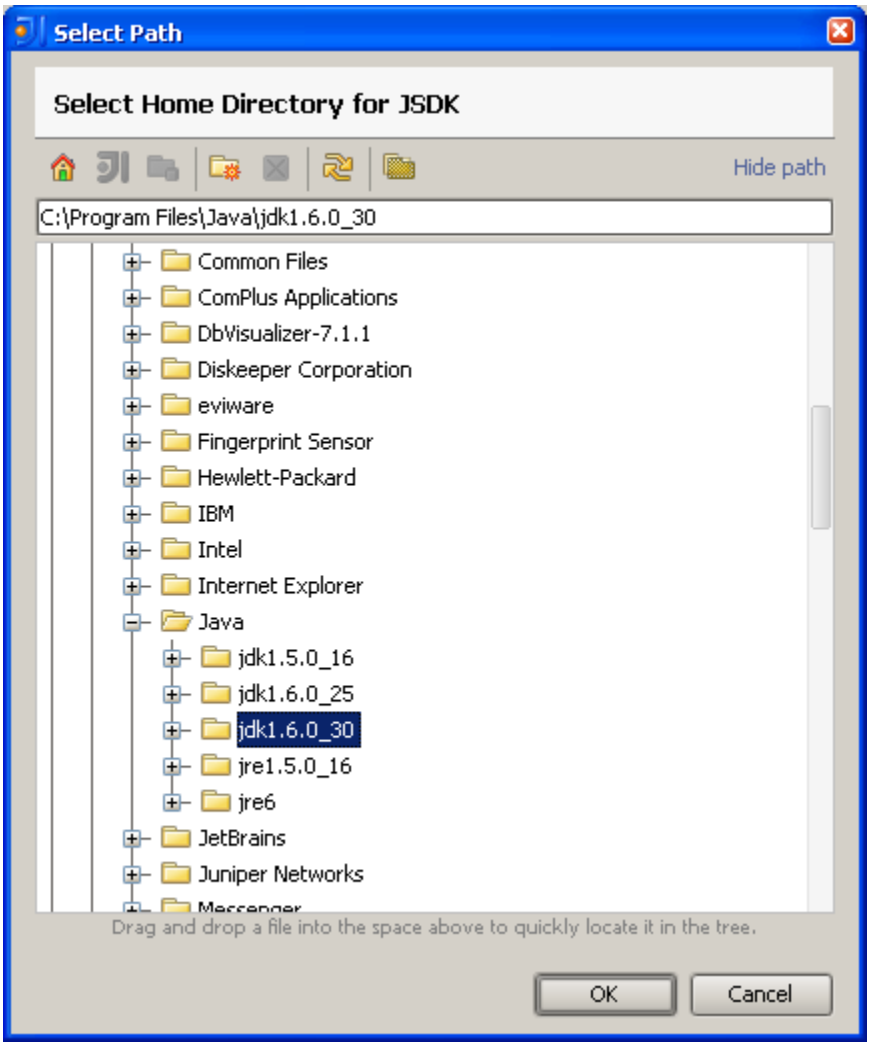
4. **JDK 1.6:** UPM 3 uses jdk 1.6.

    a. Download and install the Java SE 6 JDK from the shared drive: \\msp00100\shared\Software\UPM3

    b. Configure IntelliJ to use the 1.6 SDK. Open the Project Structure dialog (File -> Project Structure) and select the “Project” section. Under Project SDK, select “New”, then “JSDK”:



    c. Navigate to the jdk installation. For XP, typically “C:\Program Files\Java\jdk1.6.0_31”. For 64-bit Windows 7, “C:\Program Files (x86)\Java\jdk1.6.0_31”.



5. **Update to Java unlimited strength jurisdiction policy files**:
   You can download the new policy files from the Java website, or if you have access they have been added to the shared drive \\msp00100\shared\Software\java1.6\ download the jce_policy-6.zip file. inside that zip is a README with instructions on how to install. If you do not install the unlimited strength jurisdiction policy files you may see "`java.security.InvalidKeyException: Illegal key size or default parameters`" exceptions thrown. Those mean you are still running with the limited strength policies.

6. **Apache Felix:** For running the project locally, we use Apache Felix version 4.0.2. Download the .zip of the Apache Felix 4.0.2 distribution from the shared drive: \\msp00100\shared\Software\UPM3

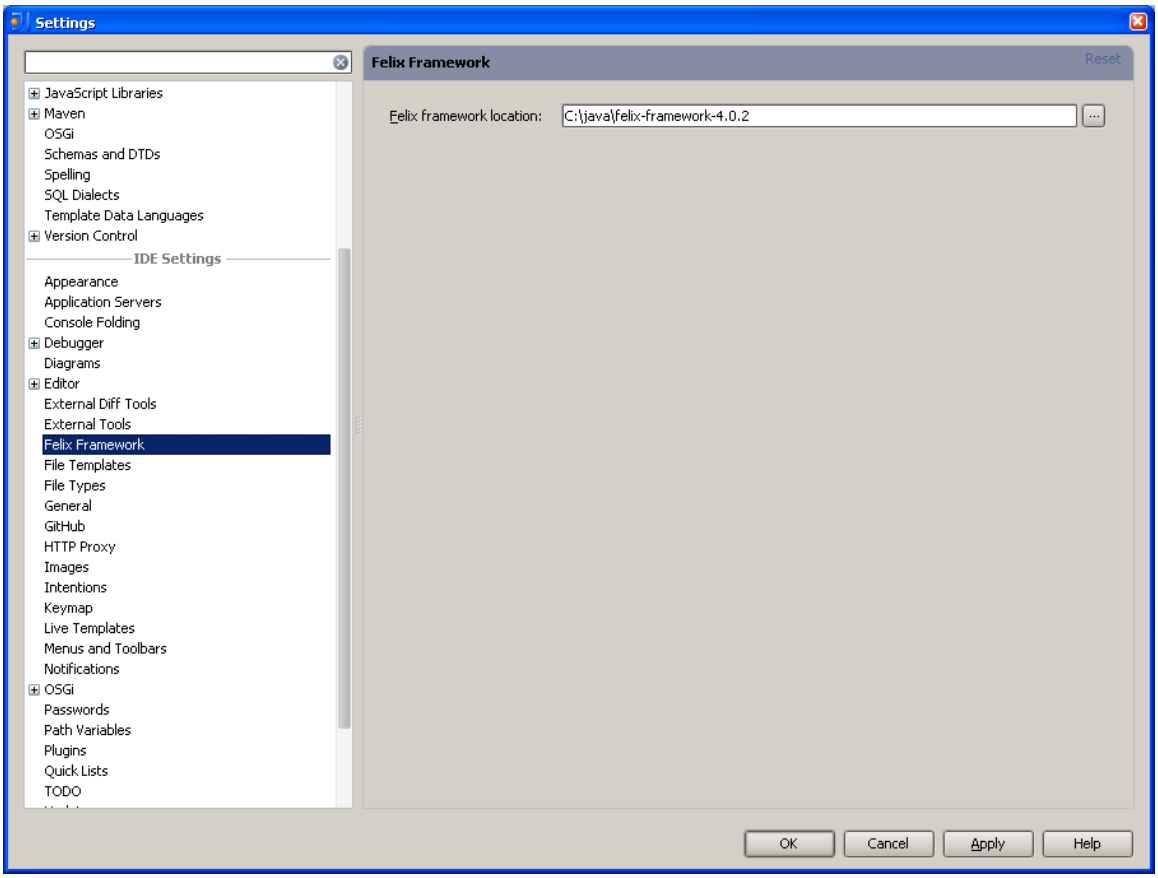
   Unzip the installation to your C: drive (e.g., C:\java).

7. **Apache Felix Plugin:** To support running the project locally in an OSGi container, we use the Apache Felix 3rd party plugin.

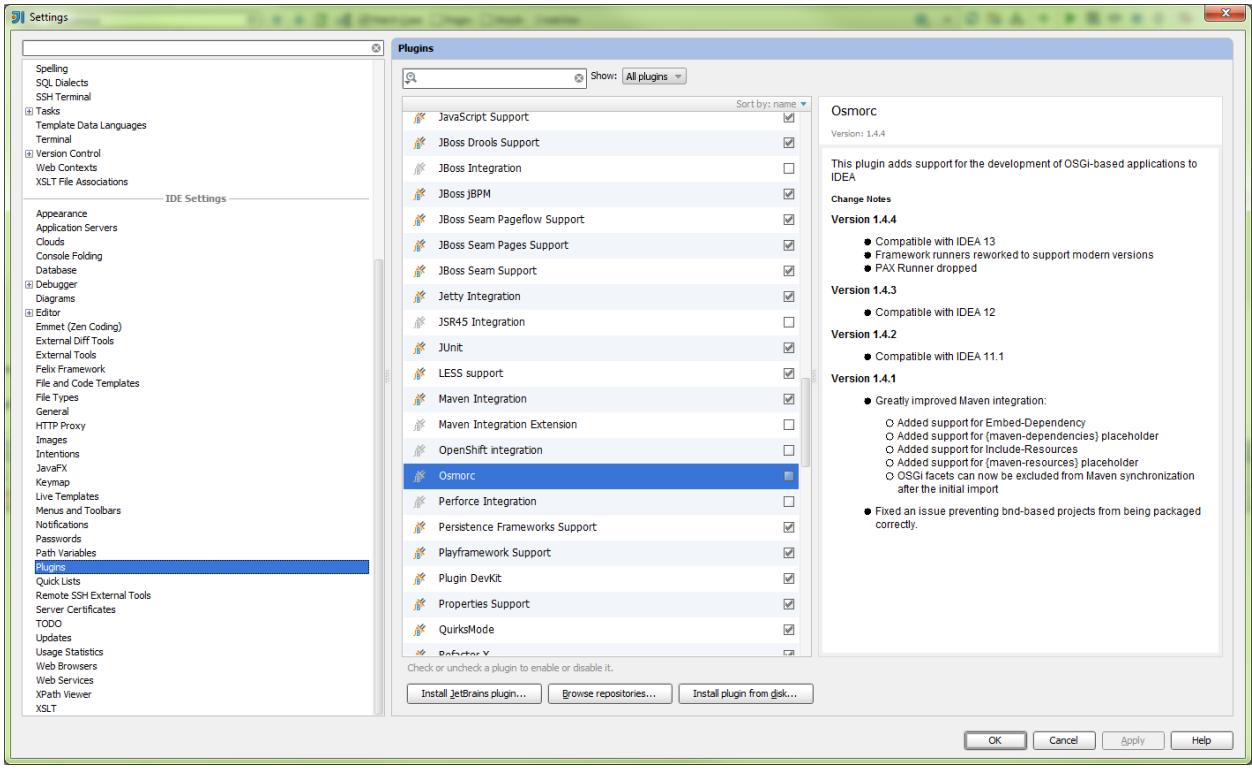
    a. Exit IntelliJ and get the apache-felix-plugin.zip from: \\msp00100\shared\Software\UPM3

    b. Unzip the apache-felix-plugin directory into: C:\Users\{your username}\.IntelliJIdea13\config\plugins\

    c. Start IntelliJ and open the settings and select “Felix Framework. Specify the Felix framework location as where you unzipped the Apache Felix 4.0.2 zip:



    d. Ensure that the “Osmorc” plugin is disabled. This is an alternative OSGi plugin that we are not using (the felix one is better). If enabled, it will try to change the project files and build extra artifacts upon compiling and possibly conflict with the Felix Plugin. You can disable it in settings under Plugins.

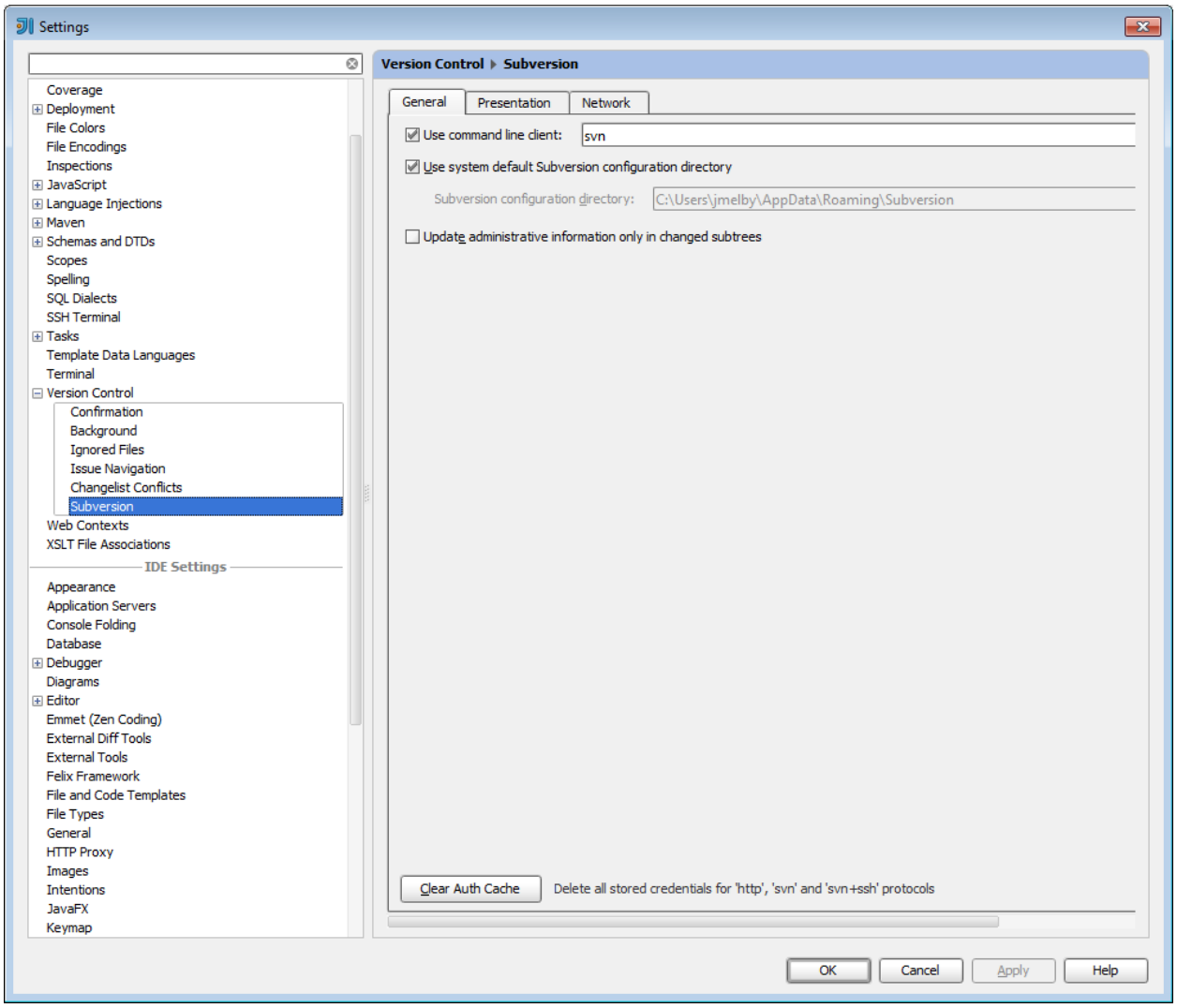


8. **UPM 3 Cryptography Project**
   You will need to checkout the keystore project form svn in order to be able to decrypt encrypted database data elements and to connect to the new MQ Managers v7.

   Checkout http://svn.uhc.com/svn/uhgit_upm3/keystore/trunk into C:\projects\keystore

9. **Felix Project**
   You will need to check out the felix project from svn to get the latest felix.properties file use for running locally.
   Checkout http://svn.uhc.com/svn/uhgit_upm3/felix/trunk into C:\projects\upm3\felix

10. **Subversion settings:**
    In order to ensure subversion works correctly with the new Tortoise 1.8 client, make sure the settings are set to use the command line client. Check the box for that setting in the Version Control/Subversion settings if it isn't already.



11. UPM 3 Project:

a. UPM3 projects are currently separated under two subversion repositories – old (i) and new (ii). The services under old subversion projects are being converted to the new single bundle projects. So anytime a service or dataservice under old project is to be modified, convert them to the new single bundle project (see conversion document). UPM3 projects under both repositories should be checked out and setup, so all services can be run.

i. This is the older repository containing UPM3 projects and here are 3 svn urls for these projects:
http://svn.uhc.com/svn/uht_upm/upm3-framework/trunk
http://svn.uhc.com/svn/uht_upm/upm3-service/trunk
http://svn.uhc.com/svn/uht_upm/upm3-dataservice/trunk

upm3-framework holds the core framework code.

upm3-service holds all of the aggregate service modules.
upm3-dataservice holds all of the dataservice modules.

It is strongly recommended to use the same local file structure so that you will not need to make changes to the felix configuration:
C:\projects\upm3-framework\trunk
C:\projects\upm3-service\trunk
C:\projects\upm3-dataservice\trunk

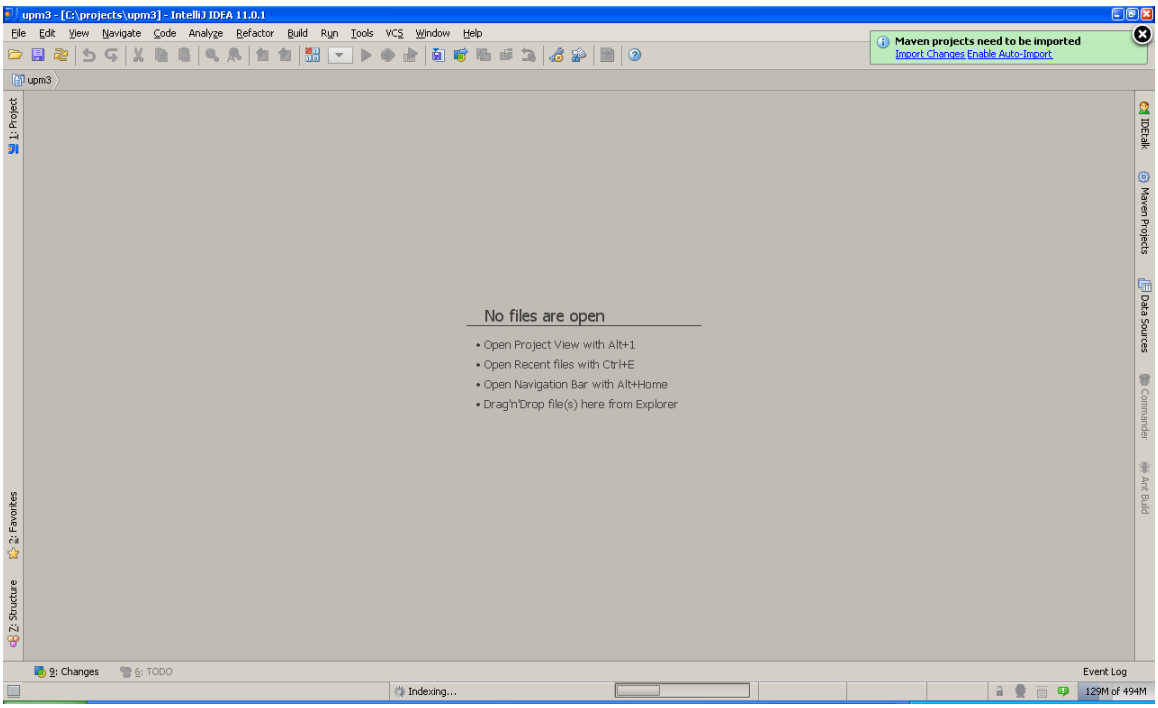
upm3-service and upm3-dataservice are organized so that each group is a separate IntelliJ project. You can check out upm3-service as a whole, or just checkout the grouping you need to work on. For instance, http://svn.uhc.com/svn/uht_upm/upm3-service/trunk will checkout all of the service groups, each group is a separate IntelliJ project. Or you can just check out http://svn.uhc.com/svn/uht_upm/upm3-service/trunk/unetmember to just work on UnetMember aggregate services.

upm3-framework is all under 1 IntelliJ project.

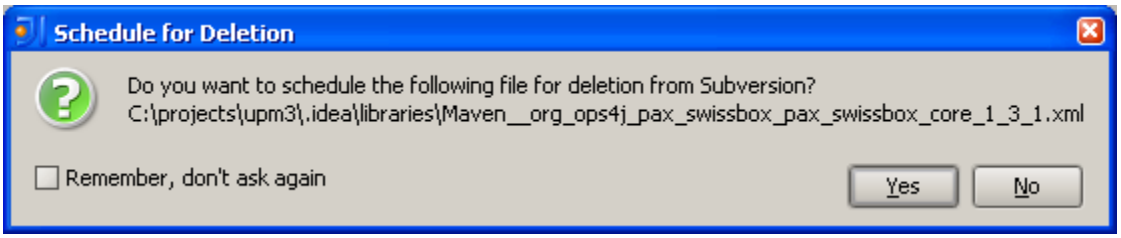
ii. This is the newer UPM3 repository - http://svn.uhc.com/svn/uhgit_upm3/.
A separate svn project is created at the root layer of this repository for each dataservice, services and framework. The service/dataservice versions are created as branches under these projects. You can either checkout the complete repository or individual projects under this repository. Any new projects should be created under this repository and not under the old repository under i.

b. Open the project in IntelliJ. Note that the UPM 3 project uses the new IntelliJ “directory” project file structure. So, instead of an .ipr file, there is an .idea directory. Instead of selecting the .ipr file, you now just select the directory where the project is checked out.

c. When the project is first opened, you will see a green dialog in the upper right of the screen notifying you that “Maven projects need to be imported”. Select the option to “Enable auto-import” so you don’t need to worry about this in the future. Note that the first time you do this, it will take a long time where it is “Resolving dependencies”. This is downloading all of the dependent libraries to your local maven repository—similar to svn-get-libs in UPM 2, but this will only need to occur once for each library on a given computer—the jar will be stored in a common location for all projects.

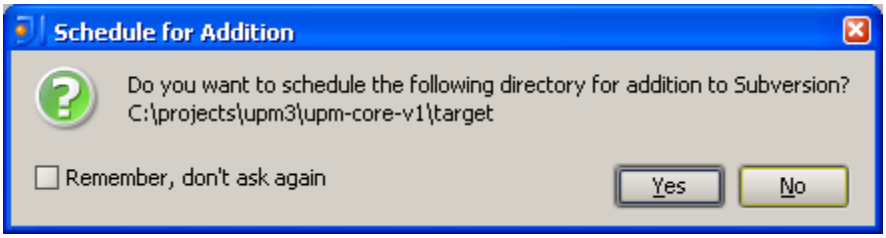


d. If you click away and come back to IntelliJ while the dependency resolution is in progress, IntelliJ may prompt you to delete library configuration files. Select "**No**" if any of these dialogs show up. IntelliJ is trying to synchronize the project libraries with the maven setup, and while it is in the middle of the resolution, it cannot determine if the library is used or not (they should disable this when in the middle of dependency resolution):

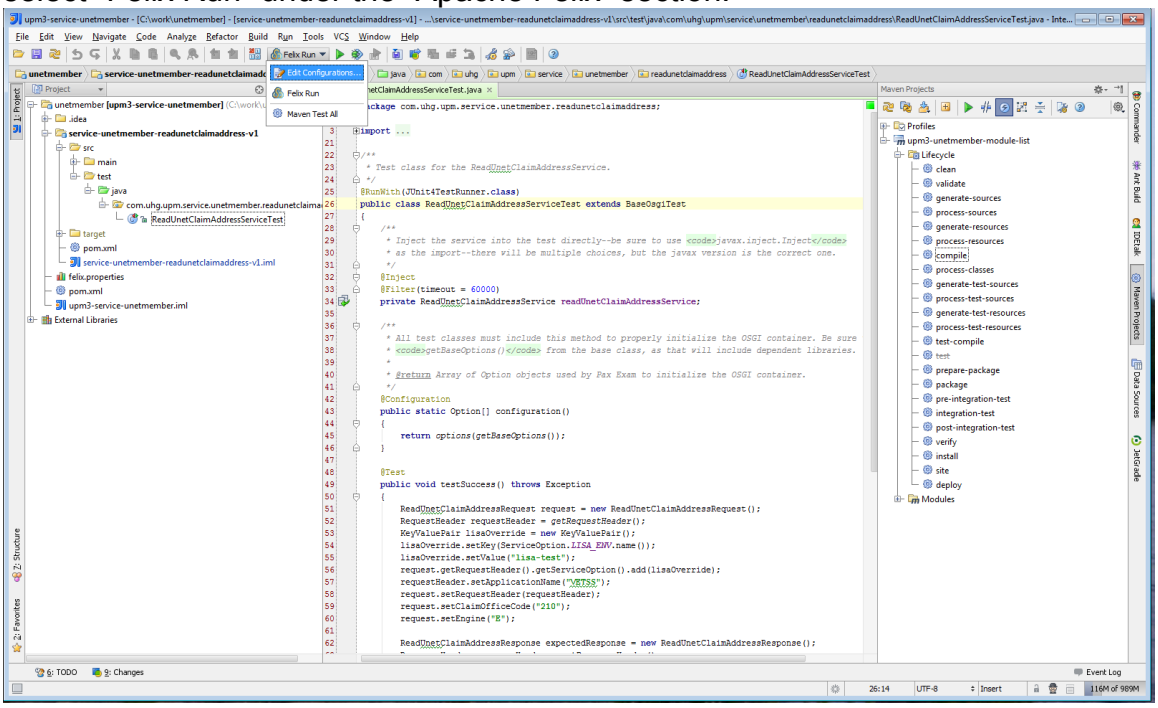


e. Let IntelliJ finish the dependency resolution and "Importing Maven projects" steps before continuing.

f. If IntelliJ asks you to add any "target" directories to Subversion, select "**No**". These are the local compiled output directories for maven and should not go in version control:

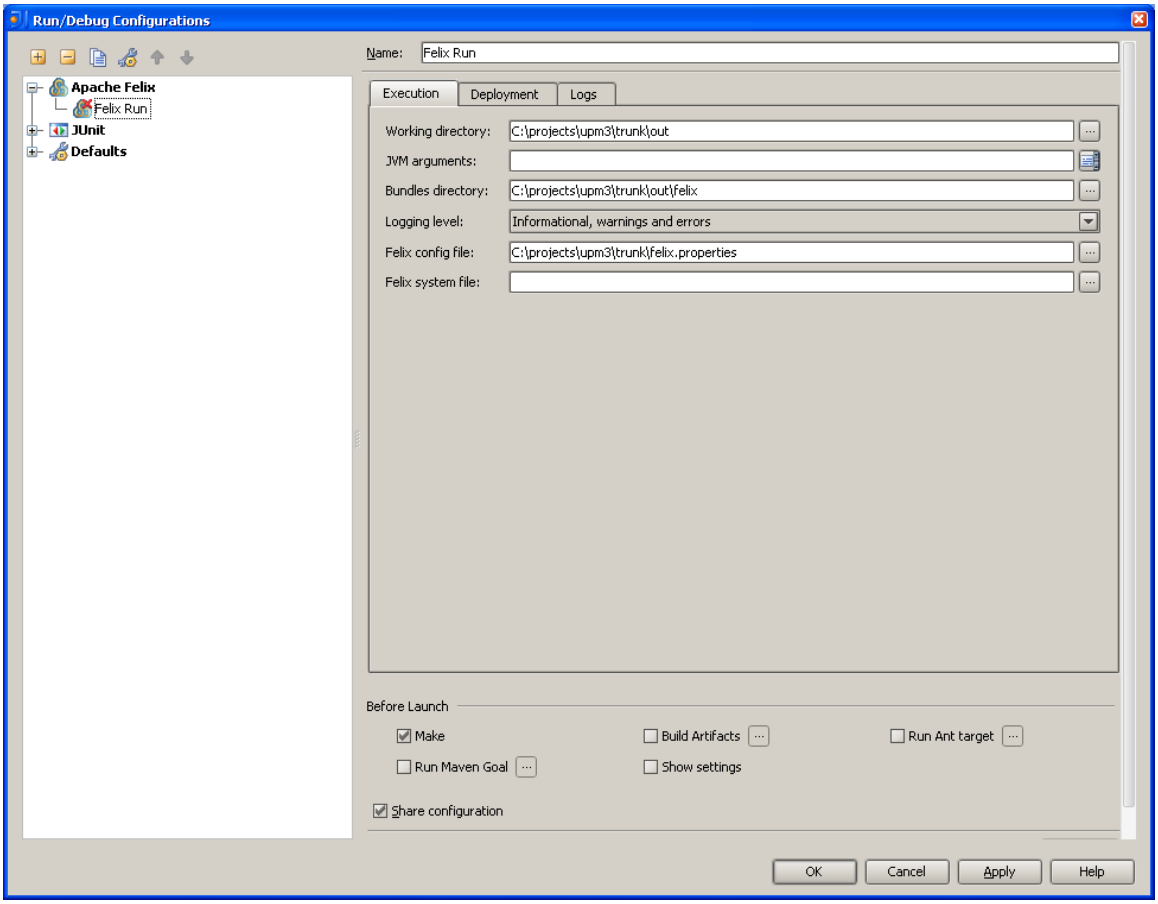


g. The “Felix Run” configuration does not use relative paths, so it must be updated to reflect the local location of where the project is checked out. Select “Edit Configurations” from the run menu, then select “Felix Run” under the “Apache Felix” section:



h. On the “Execution” tab, change the “Working directory”, “Bundles directory”, and “Felix config file” paths to your own project directory with the specified suffix paths (Note you will need to do this for every project when you check it out):

Working directory = {your project home}\out
Bundles directory = {your project home}\out\felix
Felix config file = C:\projects\upm3\felix\felix.properties



On The JVM arguments add the following parameters:
-DUPM_ENVIRONMENT=dev -DAUTH_ENABLED=false -DCRYPTOGRAPHY_PROPERTIES=/projects/keystore/upm cryptography.properties -DJNDI_MQ_STATUS=JNDI_MQ_OFF

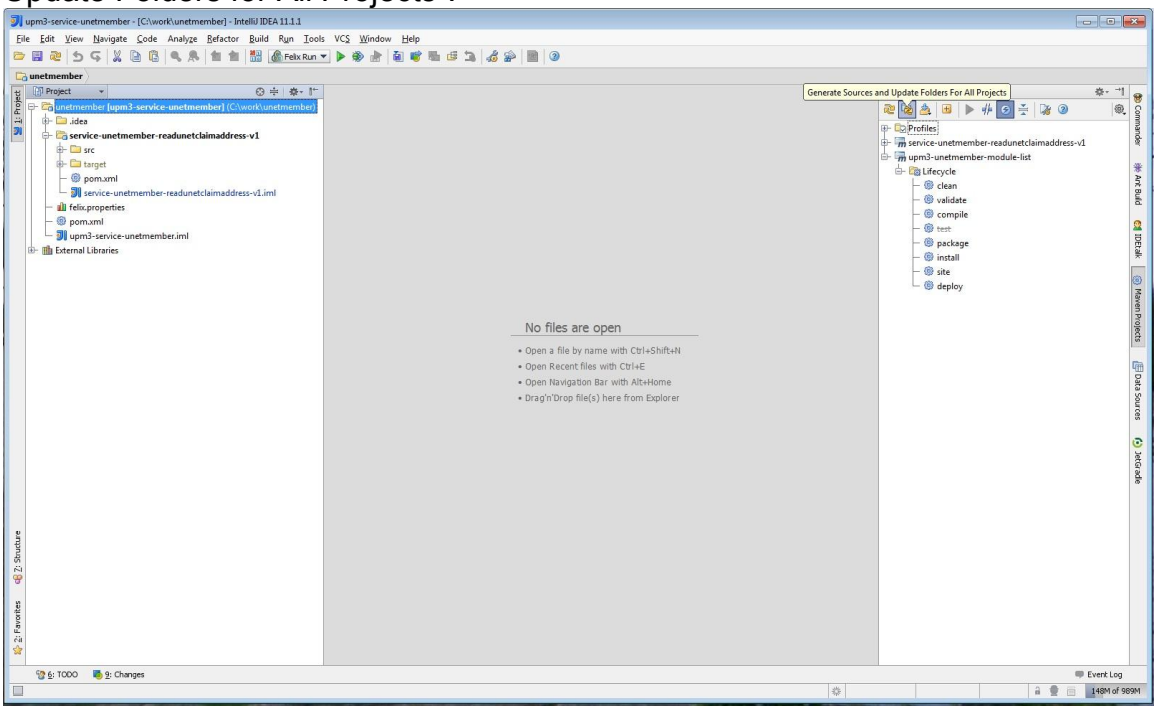
Where,
-DUPM_ENVIRONMENT is the environment you want to point. The following are the current valid values:

Environment	U3C	Legacy (They still can be used)
Development	dev	n/a
Unit Test	unitdb	n/a
Devint	devint	u3c-dev
Systest	devapp	u3c-tst
Alpha	uata	u3c-ts4
Bravo	uatb	u3c-ts2
Charlie	uatc	u3c-ts5
Master	master	u3c-ts3
Zulu	zulu	u3c-ts6

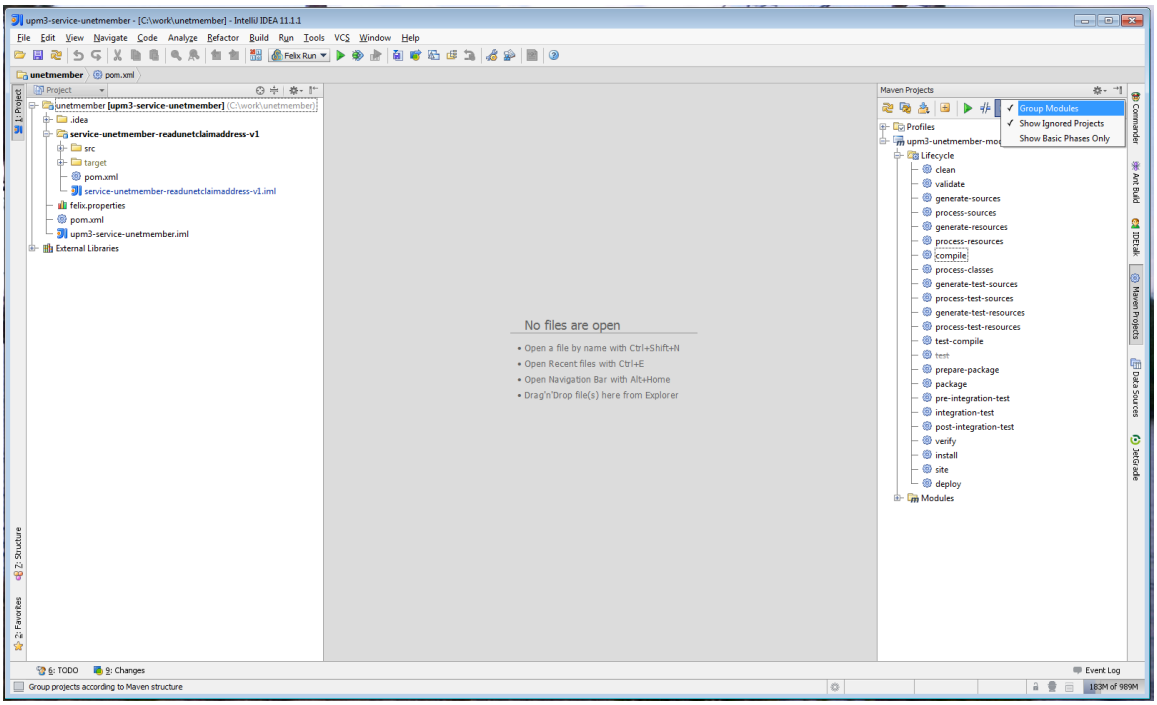
Stage	qa	n/a
Production	prod	n/a

-DAUTH_ENABLED=false. It disables the security in the incoming SOAP request.
-
DCRYPTOGRAPHY_PROPERTIES=/projects/keystore/upm cryptography.properties The location of the cryptography properties file. Please keep it in this lacation.
-DJNDI_MQ_STATUS=JNDI_MQ_OFF Do not use JNDI queue factories.

i. Before you can compile, the wsdl-to-java task must be run to generate stub code for web service back-ends. To do this, Click the button on the Maven projects tool bar, “Generate Sources and Update Folders for All Projects”.



An alternative is to expand out all of the Maven targets and run the target, to do this select the “Maven Projects” tab from the right side of the screen. In order for the task to generate sources to show up, select the gear menu in the upper right of the tab and uncheck “Show Basic Phases Only”:

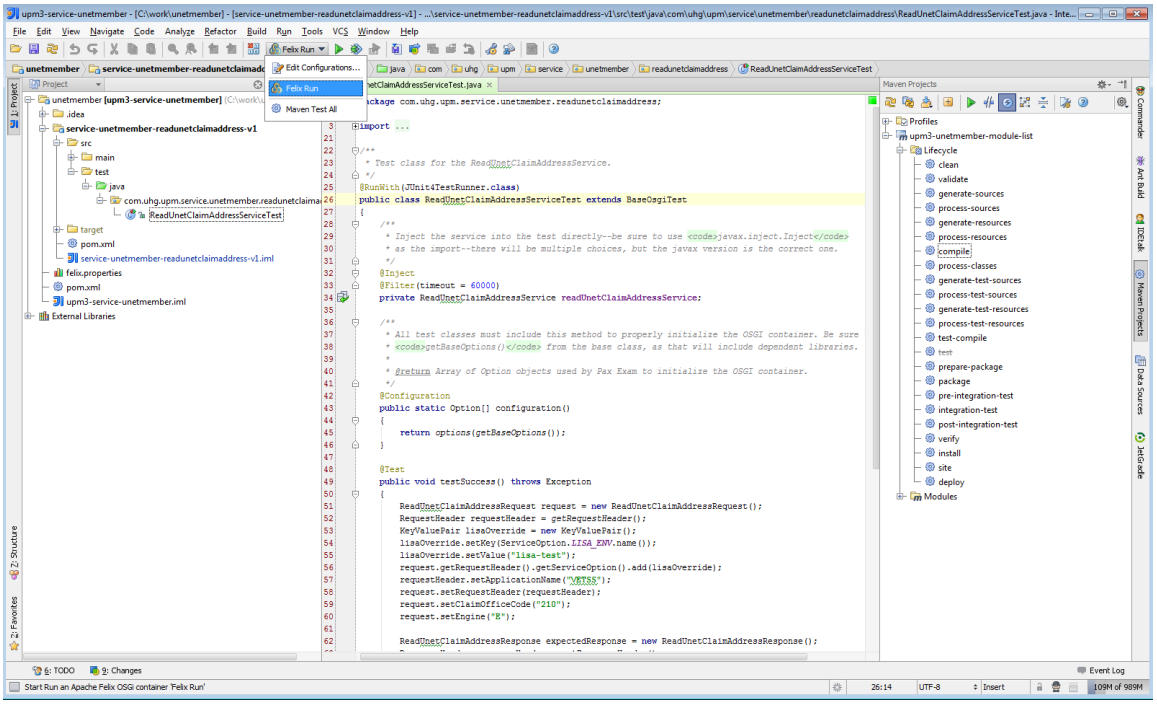


j. The first time you have checked out the project the normal generate-sources may not work because Maven will complain about project dependencies. If this happens we need to run the entire maven install to get the upm3 jar files into your local maven repository. Under “*-module-list”, expand the “Lifecycle” folder and select “install”. This will take a while the first time through.

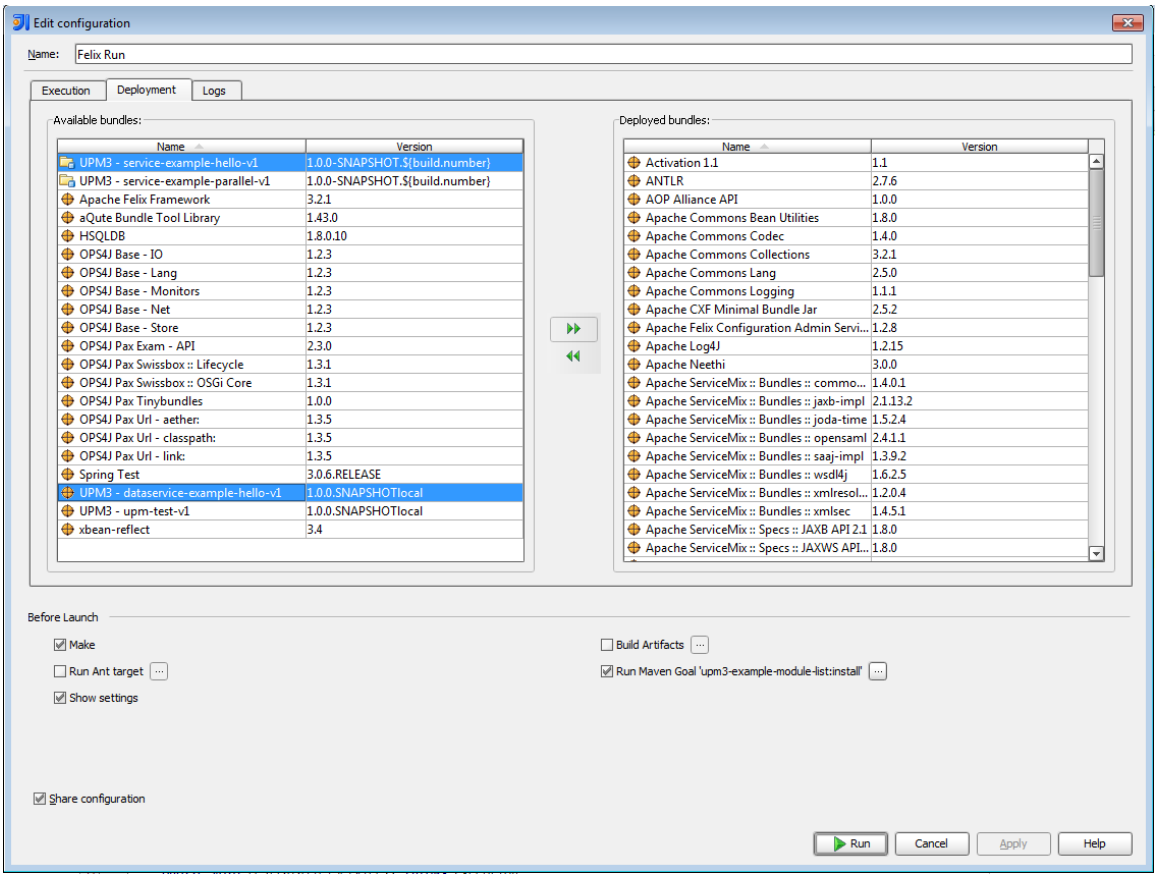
k. For future wsdl-gen changes, you should only need to click the “Generate Sources…” button or run the generate-sources target. Under “*-module-list”, expand the “Lifecycle” folder and select “generate-sources” then the green arrow to run the task to generate the stub code. This is the equivalent of the “wsdl-gen” task from UPM 2.

l. Compile the code using the make project button or menu item from the Build menu and troubleshoot any compile errors.

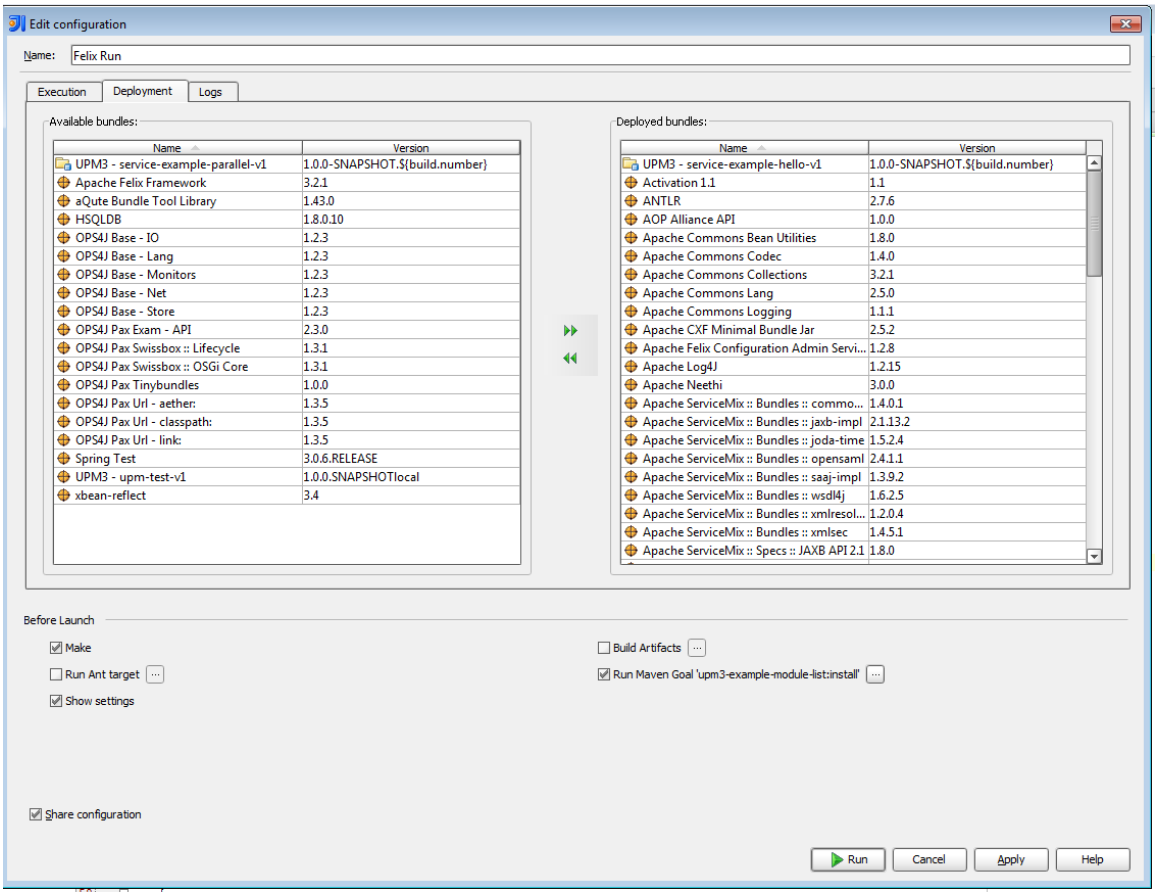
m. When you are ready to run, in UPM 3, you will choose specific bundles to run at a given time rather than the entire project. To run, select “Felix Run” from the run menu:



n. The “Felix Run” run configuration is set up with the base dependencies, but you will need to select the specific service and dataservice bundles to run. When you click run or debug, it is set to bring up the run settings. Select the Deployment tab:



o. From the left, select the service bundles you want to run ant click the arrows to put them on the right side (in this example, service-example-hello-v1 and dataservice-example-hello-v1). Take note that UPM3 bundles that are not part of this IntelliJ project but that this project is dependent on show up lower in the list and do not have the folder icon. Then click “Run”:



p. The first time you run a project, it will take longer because it is copying all of the libraries from the local maven repository. You will see output in the console, but it will not be clear when it's done starting, as there will be many message saying STARTED—one for each bundle.

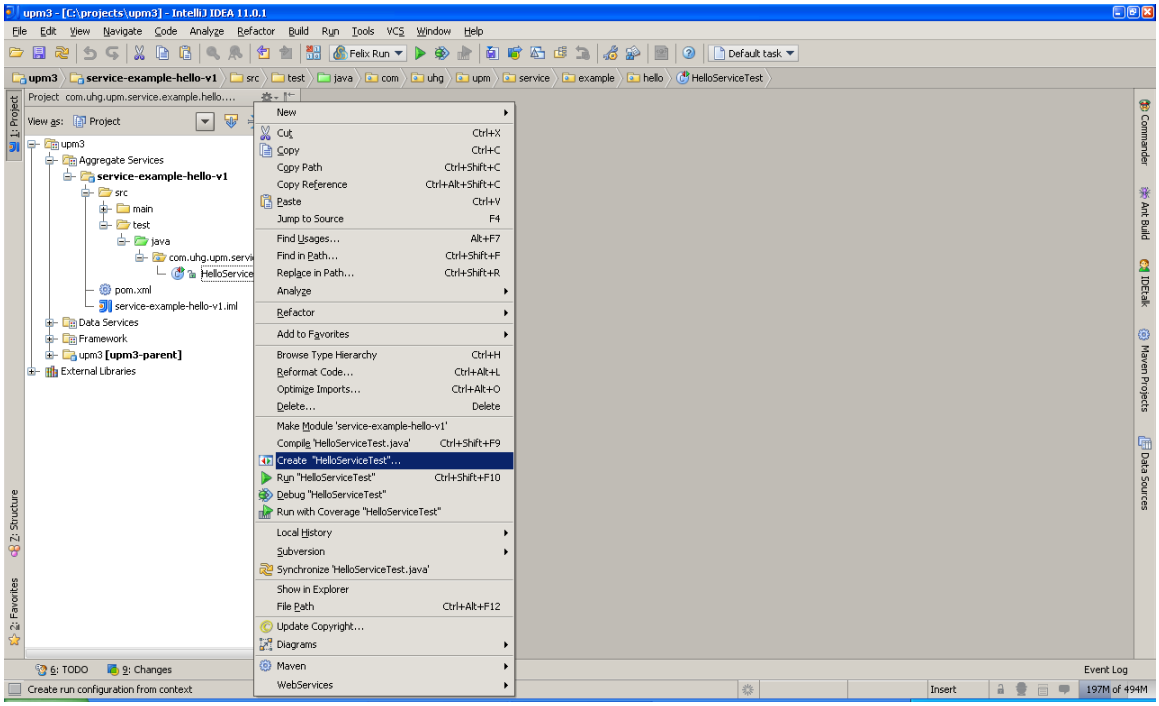
q. The first time you run a project, you may also get some errors about not being able to copy some configuration files. Just run again—these errors should go away, but if not, run another time or two.

r. When running different services, follow these same steps to remove the bundles you don't need and add the ones you do. Be sure to never check in changes to the "Felix_Run.xml" file (unless you are really updating the base configuration), as that will update the base configuration for everyone.

s. To test that everything worked, you can try running the Hello service. The WSDL for the Hello service is at: http://localhost:8080/upm3/example/HelloV1?wsdl

12. Unit Tests

a. To create a unit test run configuration, use the normal process in IntelliJ—find the test class you want to run, right click and select "Create …"



b. In UPM3 unit test classes that end with "BuildTest" in the name are automatically run in the AntHill build farm. BuildTest unit tests should always and only be pointed to a LISA virtualized service. If you need/want to have a unit test that points to a live backend you can create a unit test as normal and just name it Test. Each module will need to have a BuildTest unit test created and kept up to date, if not the build farm builds will fail.

c. Run or Debug the test as normal.

13. Development

a. If you are making a change to an existing service all that is needed is to open the group that service is in and work on your changes, if it is in the new UPM3 single bundle project (see section 8 a). If the service/dataservice is still in the old UPM3 repository (8 a i), convert it to new single bundle project and then make additional changes.

b. Use the template in Naming Convention Document to deduce the name of a new service/dataservice project being created.

c. To create a new version of a bundle follow the guidelines in UPM3 Bundle Versioning.
d. Follow instructions in AE UPM3 Project Creation when a service/dataservice is being created or modified.
e. To add a project to anthill farm, follow instructions in the document UPM3 Add Module to Build Farm.

14. Troubleshooting tools

a. If you need to get the actual endpoint that DataPower uses for a UPM CXF Data Service you can use the following REST service: http://dp-elr-stg04lo.uhc.com:20141/UPM-BACKEND/MPG/?url=upm-configure-endpoint
Where upm-configured-endpoint is what UPM has in its configuration. It will return a XML document with <BackendUrl> element.