A GUIDE ON SOAP INTEGRATION USING WS POLICY FILE

CASE STUDY : ZRA DOMESTIC TAX INTEGRATION

CELLULANT TECHNOLOGY GROUP | 14 Riverside, Cavendish Block 6th Floor

Preface

This document shows step by step integration into any soap web service that utilises policy files to sign and encrypt messages

Date	Author	Signature
2nd March 2015	Tim Mayabi	

Table of contents

INTRODUCTION TO WEB SERVICE POLICY

WS-Policy provides flexible and extensible grammar for expressing the capabilities , requirements and general characteristics of Web service entities by defining a model to express these properties as policies. This statement can be summarized by breaking it down into WS-Policy goal and what it actually specifies.

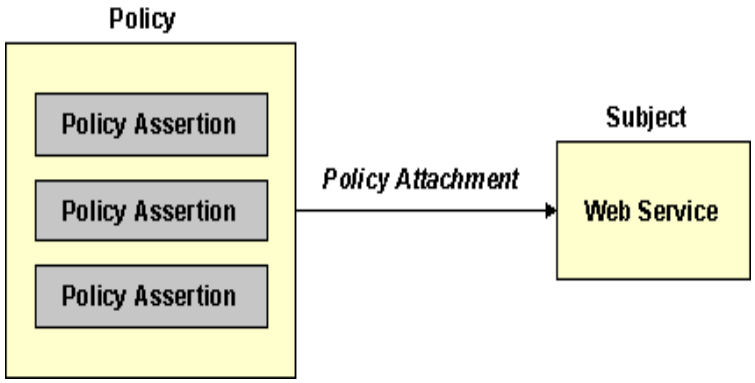
Goal:*Provides the mechanisms needed to enable web services applications to specify policies*

WS-Policy specifies:

1. An XML-based structure called a policy expression containing policy information
2. Grammar elements to indicate how the contained policy assertions apply

TERMINOLOGIES

Policy : Referes to the set of information being expressed as policy assertions
Policy Assertion : Represents an individual preference, requirement, capability etc.
Policy Expression : Set of one or more policy assertions I.e XML representation of a policy.
Policy Subject : an entity to which a policy expression can be bound
Policy Attachment : The mechanism for associating policy expressions with one or more



With the above knowledge we can go ahead and look at policy namespaces, I.e WS-Policy schema that can be used in a policy expression.

Policy Namespaces

Prefix	Description	Namespace
wsp	WS-Policy,WS-PolicyAssertions and WS-Policy Attachment	http://schemas.xmlsoap.org/ws/2002/12/policy
wsse	WS-Security Policy	http://schemas.xmlsoap.org/ws/2002/12/secext
wsu	WS utility schema	http://schemas.xmlsoap.org/ws/2002/07/utility
msp	WSE 2.0 policy schema	http://schemas.microsoft.com/wse/2003/06/Policy

<wsp:Policy xmlns:wsp="..." xmlns:wsu="..." wsu:Id="..." Name="..." TargetNamespace="...">

```
        <!-- policy assertions go here -->

    </wsp:Policy>
```

The wsu:Id attribute assigns the policy expression an ID value as a URI

Policy Assertion.

Architecture

```
<wsp:Policy xmlns:wsp="..." xmlns:wsu="..." wsu:Id="..."        Name="..." TargetNamespace="..." >


      <Assertion wsp:Usage="..." wsp:Preference="..." />
     <Assertion wsp:Usage="..." wsp:Preference="..." />
  ... </wsp:Policy>
```

AssertionExample

```
<wsp:Policy xmlns:wsp="..." xmlns:wsse="...">

  <wsse:SecurityToken wsp:Usage="wsp:Required">

<wsse:TokenType>wsse:Kerberosv5ST</wsse:TokenType>

  </wsse:SecurityToken>

  <wsse:Integrity wsp:Usage="wsp:Required">

    <wsse:Algorithm Type="wsse:AlgSignature“

      URI="http://www.w3.org/2000/09/xmlenc#aes" />

  </wsse:Integrity>

</wsp:Policy>
```

The above example specifies two policy states I.e :

- Security token is required
- Use of AES is required

AssertionPreference

wsp:Preference attribute:

Used to specify the service’s preference as an integer value

Larger integer => higher preference

Omitted preference attribute is interpreted as a 0

```
<wsp:Policy xmlns:wsp="..." xmlns:wsse="...">
  <wsse:SecurityToken wsp:Usage="wsp:Optional">
    <wsse:TokenType>wsse:UsernameToken</wsse:TokenType>
  </wsse:SecurityToken>
  <wsse:SecurityToken wsp:Usage="wsp:Optional“
    wsp:Preference="1">
    <wsse:TokenType>wsse:x509v3</wsse:TokenType>
  </wsse:SecurityToken>
</wsp:Policy>
```

This assertion state means that the subject prefers X.509 certificates over username tokens.

Standard Policy Assertions.

This defines four general policy assertions for any subject

Policy Assertion	Description
wsp:TextEncoding	Specifies a character encoding
wsp:Language	Specifies a natural language (xml:Lang)
wsp:SpecVersion	Specifies a version of a particular specification
wsp:MessagePredicate	Specifies a predicate that can be tested against the message (XPath expressions by default)

```
<wsp:Policy xmlns:wsse="...">
  <wsp:TextEncoding wsp:Usage="wsp:Required" Encoding="utf-8"/>
  <wsp:Language wsp:Usage="wsp:Required" Language="en"/>
  <wsp:SpecVersion wsp:Usage="wsp:Required"
    URI="http://www.w3.org/TR/2000/NOTE-SOAP-20000508/" />
  ...
</wsp:Policy>
```

Therefore the subject requires:

- The UTF-8 character encoding

- Any form of the English language

- SOAP version 1.1

Policy Reference

Mechanism to share policy assertions across policy expressions

```
<wsp:Policy xmlns:wsp="...">

  ...

  <wsp:PolicyReference URI="..."

                       Ref="..."

               Digest="..."

               DigestAlgorithm="..." />

  ...

</wsp:Policy>
```

Policy reference example.

```
<wsp:Policy wsu:Id="tokens" xmlns:wsp="..." xmlns:wsse="...">

  <wsp:ExactlyOne wsp:Usage="Required">

    <wsse:SecurityToken>

      <wsse:TokenType>wsse:UsernameToken</wsse:TokenType>

    </wsse:SecurityToken>

    <wsse:SecurityToken wsp:Preference="10">

      <wsse:TokenType>wsse:x509v3</wsse:TokenType>

    </wsse:SecurityToken>

    <wsse:SecurityToken wsp:Preference="1">
```

```
 <wsse:TokenType>wsse:Kerberosv5ST</wsse:TokenType>

   </wsse:SecurityToken>

  </wsp:ExactlyOne>

</wsp:Policy>
```

Visit this site http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnwebsrv/html/understwspol.asp to learn more about ws-policy.

With the knowledge on ws-policy and security we will go ahead and install the following as components required in this guide.

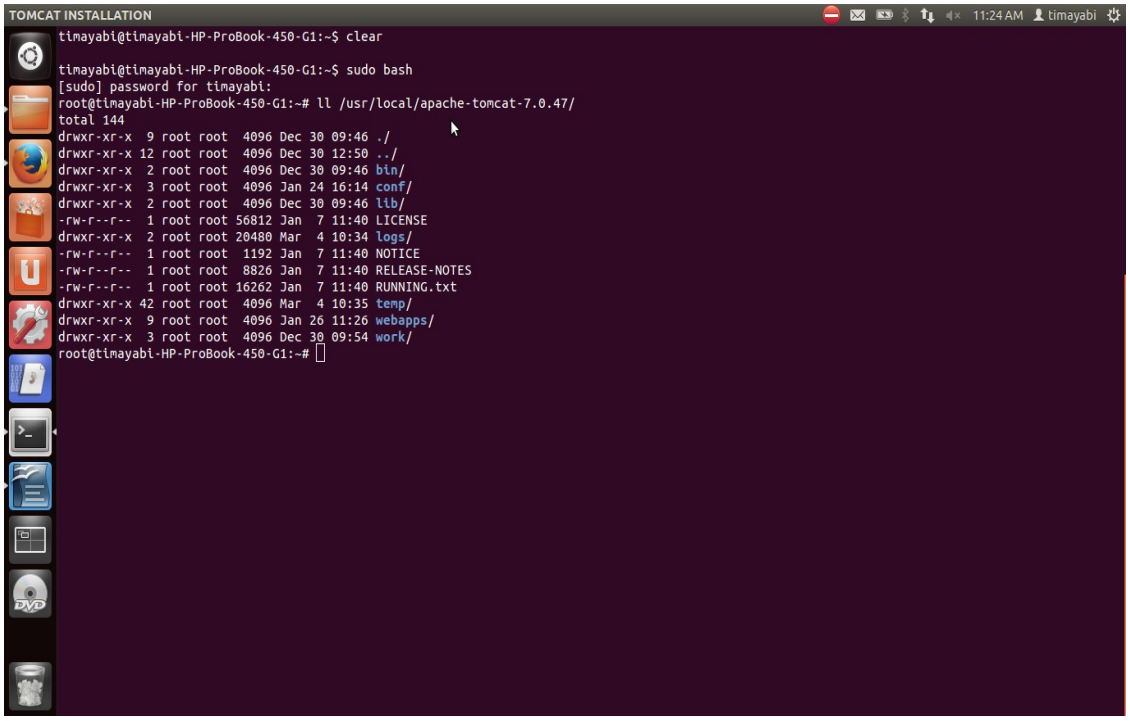
1. Tomcat 7
2. Axis2
3. Ramapart and Rahas axis 2 security modules.
4. IntelliJ IDEA IDE.
5. wsdl2php
6. Soap UI

TOMCAT, AXIS 2 AND RAMPART SECURITY MODULE INSTALLATION STEPS

Tomcat Installation

In this guide ubuntu 12.04 LTS was used and below are the steps on how to install Tomcat 7.

1. Open terminal
2. Type this command 'sudo apt-get install tomcat7'
3. Enter your password
4. Tomcat will be installed in the '/usr/local/apache-tomcat-7.0.47/' path as shown in the diagram below



5. Tomcat start – stop script will be on '/etc/init.d/' path with the name tomcat747
6. Check if java is installed by typing on the terminal this command 'java -version'.
7. To install java type this command 'apt-get install default-jdk'
8. On the terminal change directory to your root/home folder and configure '.bashrc script'
9. Type vim .bashrc to open the script and add this text at the bottom of the script. 'JAVA_HOME=/usr/lib/jvm/java-7-oracle

' . In this guide I am using java-7-oracle. Make sure you make the change according to the java 7 version you are using.

10. To know the java path, type this command 'update-alternatives --config java

' then it will display the following

There are 3 choices for the alternative java (providing /usr/bin/java).

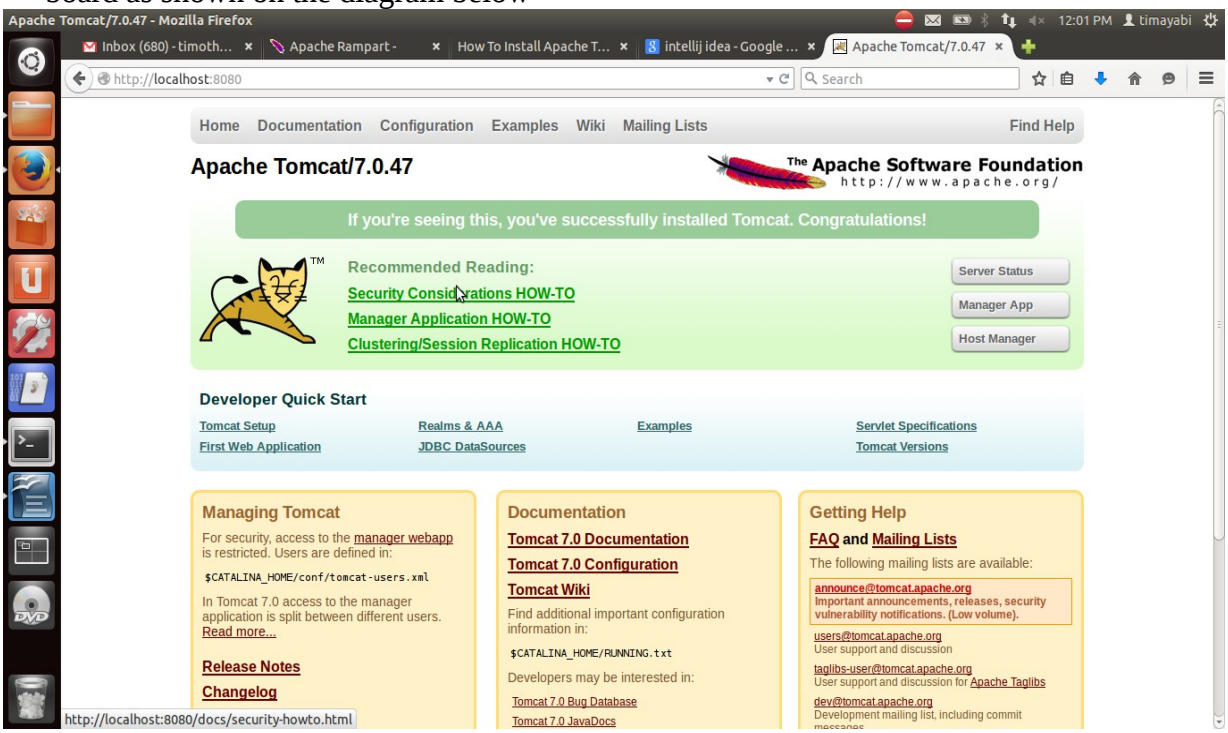
Selection Path Priority Status

```
* 0            /usr/lib/jvm/java-7-oracle/jre/bin/java          1062      auto mode
  1            /usr/lib/jvm/java-6-openjdk-amd64/jre/bin/java   1061      manual mode
  2            /usr/lib/jvm/java-7-openjdk-amd64/jre/bin/java   1051      manual mode
  3            /usr/lib/jvm/java-7-oracle/jre/bin/java          1062      manual mode
```

Press enter to keep the current choice[*], or type selection number

11. Select only java 7 version paths upto before '/jre/bin/java' and copy it to the .bashrc script where you had written 'JAVA_HOME ='. Dont forget to define and export JAVA_HOME path ie PATH=$PATH:$JAVA_HOME/bin then export PATH

12. Save file, restart your machine, open terminal and type 'echo $JAVA_HOME' just to be sure that java has been configured correctly. It will output the path to java on the terminal. eg '/usr/lib/jvm/java-7-oracle'.
13. When you type http://localhost:8080 on your browser you should be able to see the tomcat dash board as shown on the diagram below



You can define your tomcat users on 'tomcat-users.xml' located on this path '/usr/local/apache-tomcat-7.0.47/conf
', restart tomcat then login to Host Manager and Manager App with your credentials.

Axis 2 , Rampart and Rahas module installation.

Axis 2 is an apache web service engine deployed on tomcat while Rampart and Rahas are security modules suppoting WS-Security, WS-Security policy and WS-Trust.

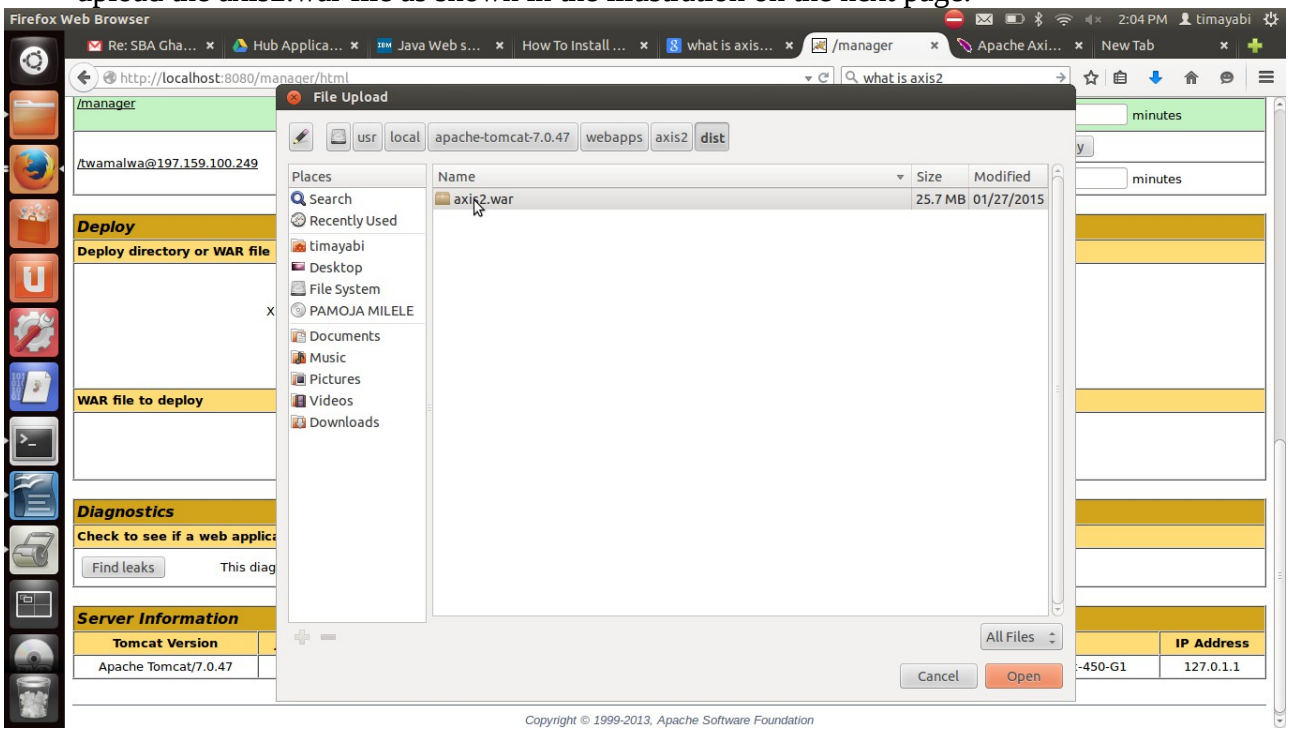
1. Download rampart 1.6.2 Standard Binary Distribution from this link. https://axis.apache.org/axis2/java/rampart/download/1.6.2/download.cgi
2. Extract it to your preferred folder and inside the folder you will see rahas-1.6.2.mar and rampart 1.6.2.mar. These modules and all the contents in the lib folder will be copied to the axis lib directory.
3. Download axis1.6.2 binary distribution from this link. http://axis.apache.org/axis2/java/core/download.cgi
4. Extract axis to this path '/usr/local/apache-tomcat-7.0.47/webapps' and rename it to 'axis2'
5. Ensure step 2 is done
6. Install ant by typing on the terminal apt-get install ant
7. While still on terminal, change directory to this path. '/usr/local/apache-tomcat-7.0.47/webapps/axis2/webapp' , ensure that the build.xml is available and run ant command by just typing ant. If build is successful the information below will be displayed.

```
init:

    [mkdir] Created dir: /usr/local/apache-tomcat-7.0.47/webapps/axis2/dist

    [mkdir] Created dir:/usr/local/apache-tomcat-7.0.47/webapps/axis2/dist/temp

     [copy] Copying 59 files to /usr/local/apache-tomcat-7.0.47/webapps/axis2/dist/temp


prepare.repo:

     [copy] Copying 9 files to /usr/local/apache-tomcat-7.0.47/webapps/axis2/dist/temp/WEB-INF

    [mkdir] Created dir: /usr/local/apache-tomcat-7.0.47/webapps/dist/temp/WEB-INF/conf

     [copy] Copying 1 file to /usr/local/apache-tomcat-7.0.47/webapps/axis2/dist/temp/WEB-INF/conf


create.war:

      [war] Building war: /usr/local/apache-tomcat-7.0.47/webapps/axis2/dist/axis2.war

   [delete] Deleting directory /usr/local/apache-tomcat-7.0.47/webapps/axis2/dist/temp


BUILD SUCCESSFUL

Total time: 1 second
```

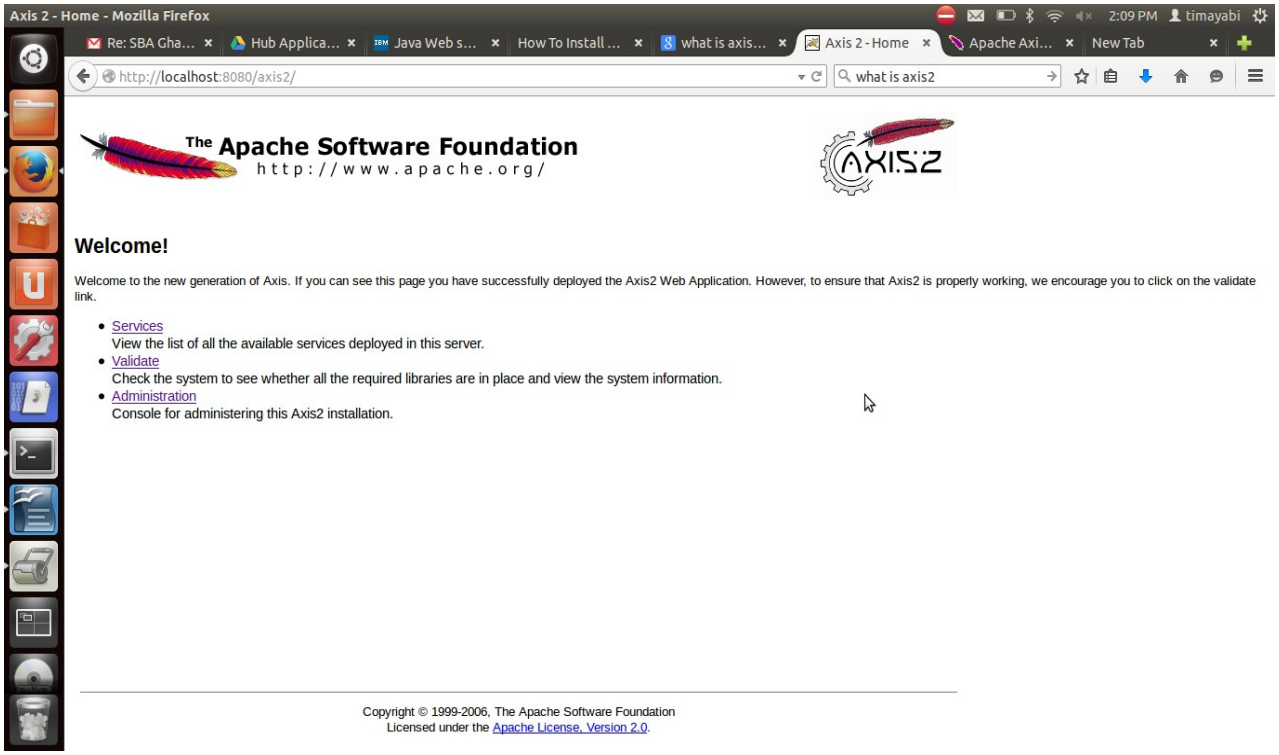
Change directory to '/usr/local/apache-tomcat-7.0.47/webapps/axis2/dist' and ensure that axis2.war is available.

8. Go to '.bashrc' script on your root folder and add AXIS_HOME I.e AXIS2_HOME=/usr/local/apache-tomcat-7.0.47/webapps/axis2. Dont forget to define and export AXIS2_HOME path ie PATH=$PATH:$JAVA_HOME/bin:$AXIS2_HOME/bin then export PATH

9. Go to the browser and login to manager App , find 'war file to deploy' , click on browse and upload the axis2.war file as shown in the illustration on the next page.



After a successful upload restart tomcat then copy and paste this link on your browser. http://localhost:8080/axis2/ You should be able to see something similar to the illustration below.

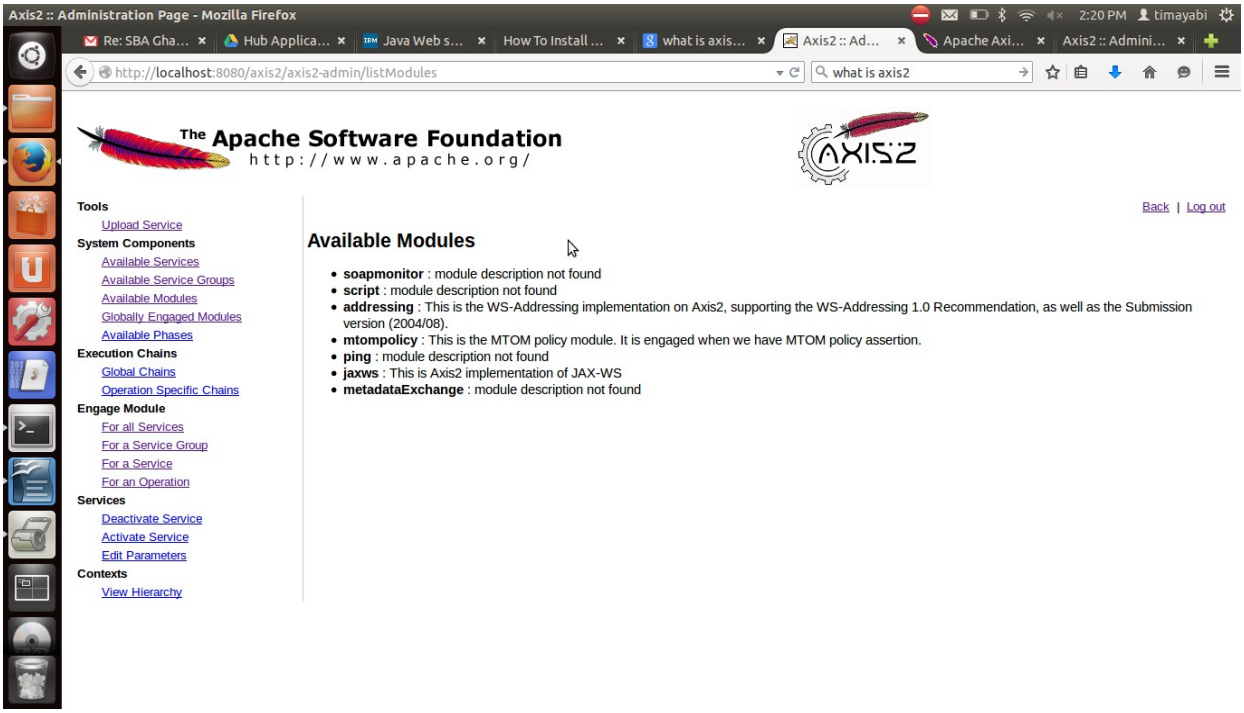


10 . Go to this directory '/usr/local/apache-tomcat-7.0.47/webapps/axis2/conf' and open axis2.xml
Inside that file you will get your axis login credentials. ie

<parameter name="userName">admin</parameter>

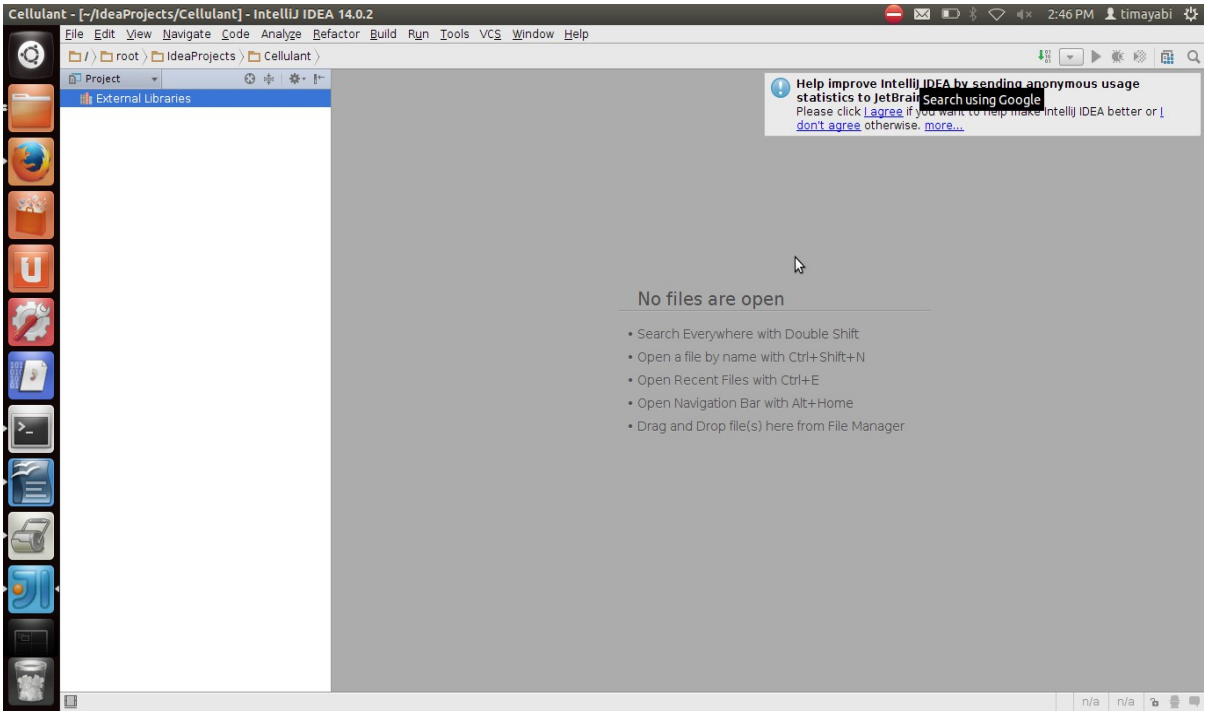
<parameter name="password">axis2</parameter>

Go to axis2 home page on your browser and click on the administration link.
Enter using the above credentials or with credentials you have configured on axis2.xml file.
You should be able to see the page as per the illustration below. Make sure addressing module is available once you click on the 'available modules ' link.


Axis2 :: Administration Page - Mozilla Firefox
2:20 PM timayabi
Re: SBA Gha…
Hub Applica…
Java Web s…
How To Install …
what is axis…
Axis2 :: Ad…
Apache Axi…
Axis2 :: Admini…
http://localhost:8080/axis2/axis2-admin/listModules
what is axis2
The Apache Software Foundation
http://www.apache.org/
AXIS2
Back | Log out
Tools
Upload Service
System Components
Available Services
Available Service Groups
Available Modules
Globally Engaged Modules
Available Phases
Execution Chains
Global Chains
Operation Specific Chains
Engage Module
For all Services
For a Service Group
For a Service
For an Operation
Services
Deactivate Service
Activate Service
Edit Parameters
Contexts
View Hierarchy
Available Modules
soapmonitor : module description not found
script : module description not found
addressing : This is the WS-Addressing implementation on Axis2, supporting the WS-Addressing 1.0 Recommendation, as well as the Submission version (2004/08).
mtompolicy : This is the MTOM policy module. It is engaged when we have MTOM policy assertion.
ping : module description not found
jaxws : This is Axis2 implementation of JAX-WS
metadataExchange : module description not found

IntelliJ IDEA installation

1. Download the latest ide from this link https://www.jetbrains.com/idea/download/index.html#linux
2. Extract it to your preferred folder, change directory to the folder, go into the bin directory and start the IDE with this command 'sh idea.sh'.



WSDL2PHP installation

1. On the terminal type this command 1. On the terminal type this command 'sudo pear install wsdl2php-0.2.1-pear.tgz'
2. If you dont have pear type this commands to install it. wget http://pear.php.net/go-pear.phar and php go-pear.phar

NB. This is applicable with php version 5.3 and above

SoapUI installation

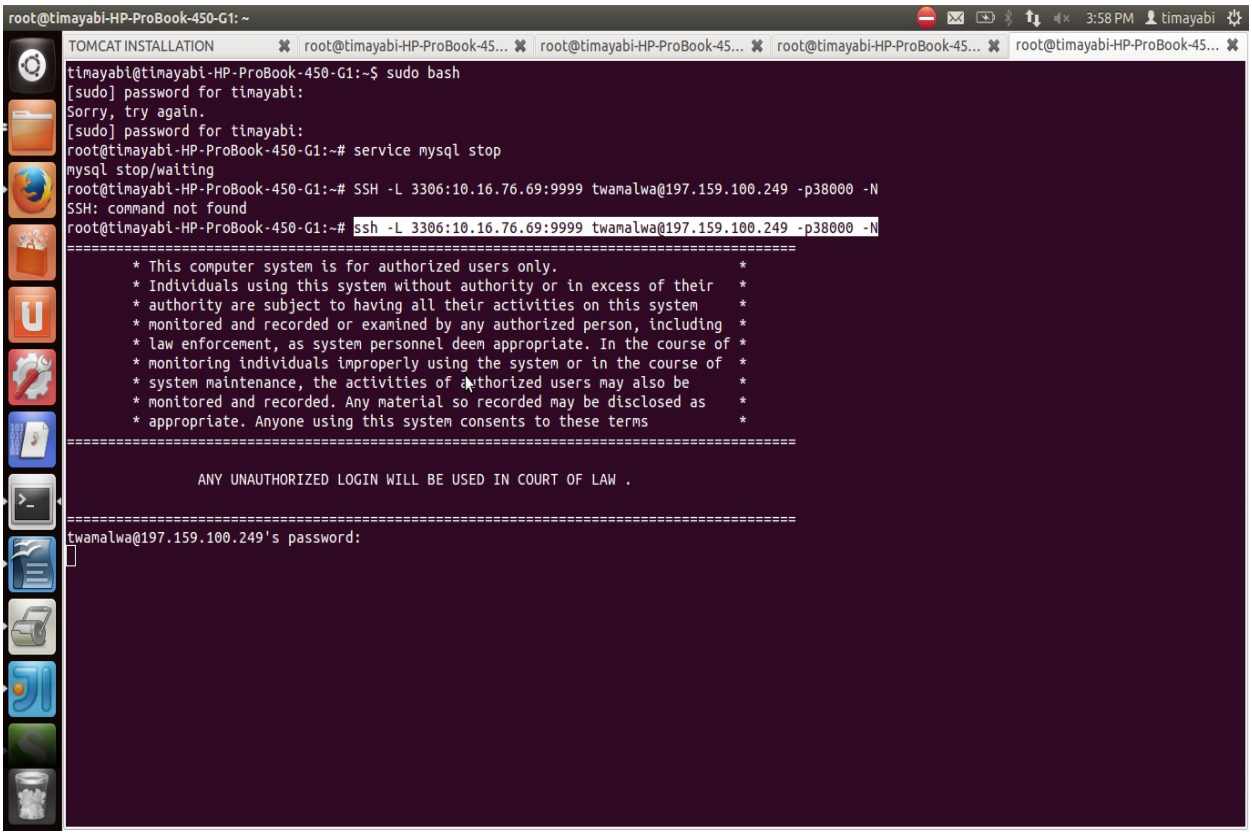
1. Download soapUI from this link http://www.soapui.org/
2. Extract the zip folder
3. Change directory to bin folder and run the sh file I.e 'sh soapui.sh' to do the installation.
4. After installation click on the 'Dash Home' icon. This icon is shown in the illustration on the next page. Focus on the mouse arrow.

5. After clicking on 'Dash Home', type 'soapui' then you will be able to see the soapUI icon and click on it to start soapUI.

Domestic Tax Integration

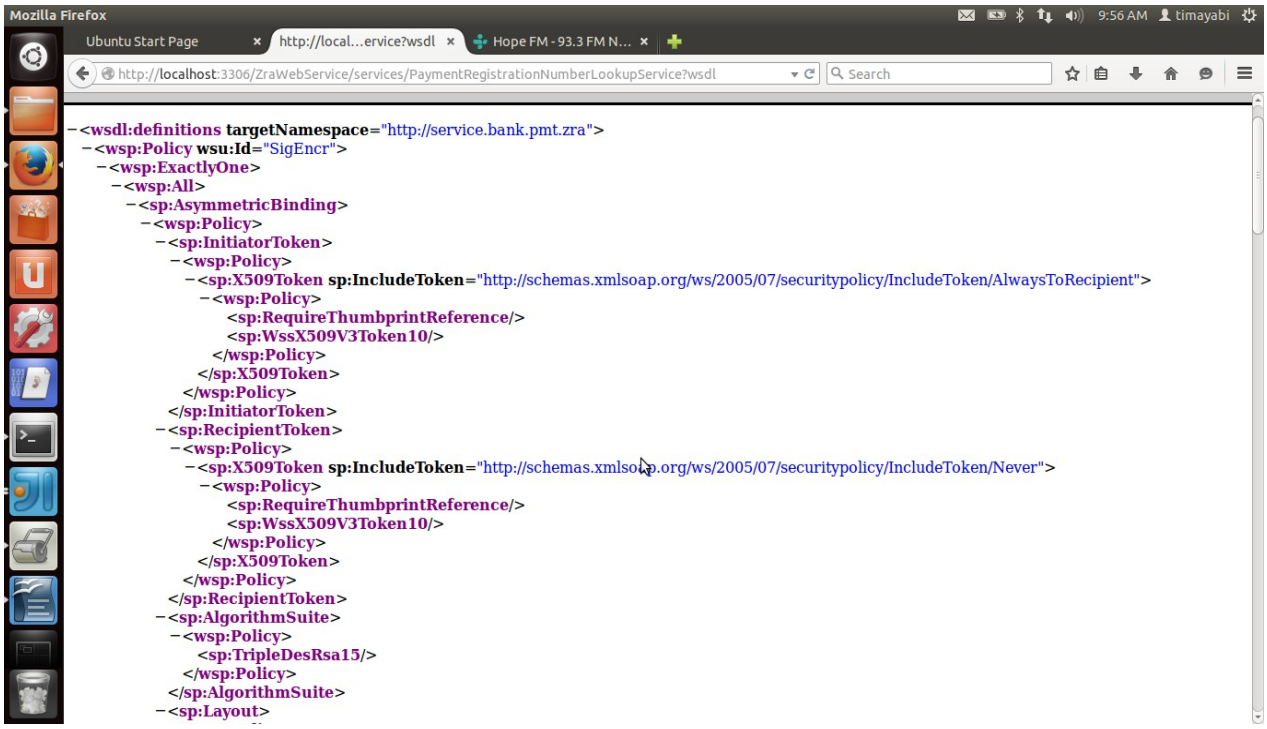
So far we have downloaded and installed all the tools required to enable us integrate to Zambia Revenue Authority Domestic Tax Services.
However apart from the documentation shared by Zambia Revenue Authority on Domestic Tax Services, it will be better for us to have a practical view of the kind of requests they expect from us. This can easily be achieved by Soap UI. Still the other problem is that the VPN set up is done on our local machine but rather on a server (In this case 197.159.100.2490-p38000) which we can access from our local machine. To be able to connect to ZRA domestic tax services via our local machine follow the steps below.

1. Open terminal.
2. This is the syntax for port fowarding 'ssh -L localport:host:hostport user@ssh_server -N' . For example I used this command 'ssh -L 3306:10.16.76.69:9999 twamalwa@197.159.100.249 -p38000 -N'. Also note, I have killed mysql running on port 3306 to free up the port. See the illustration below

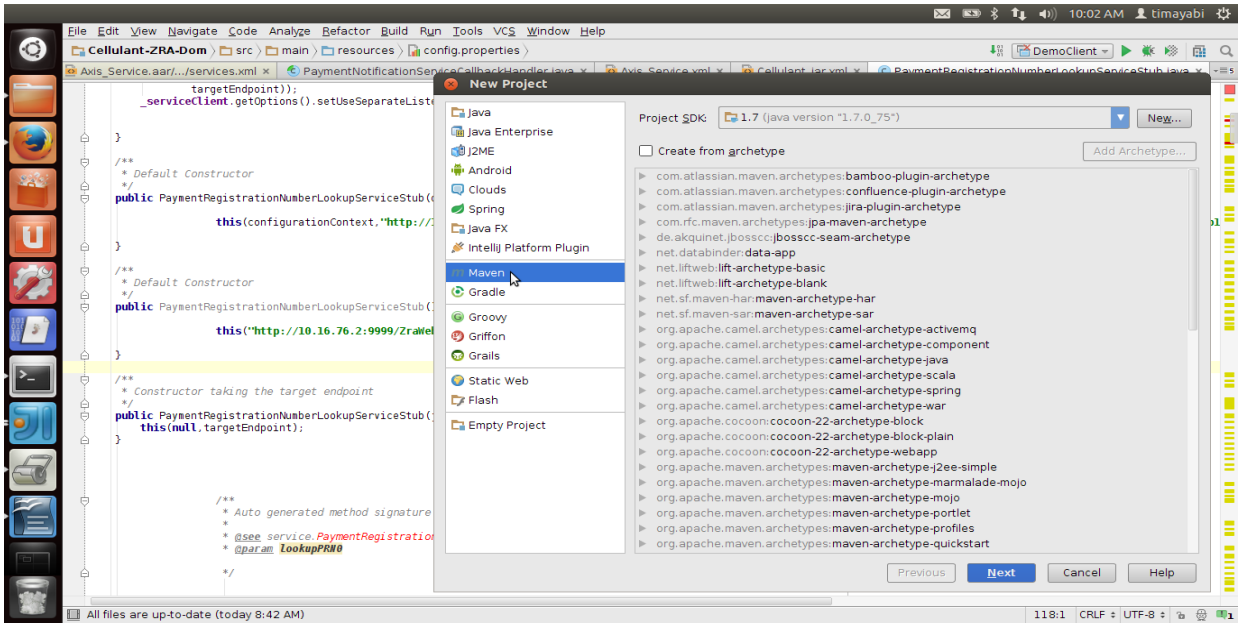


3. Enter your password and dont expect it to show anything. Thats it port fowarding is done

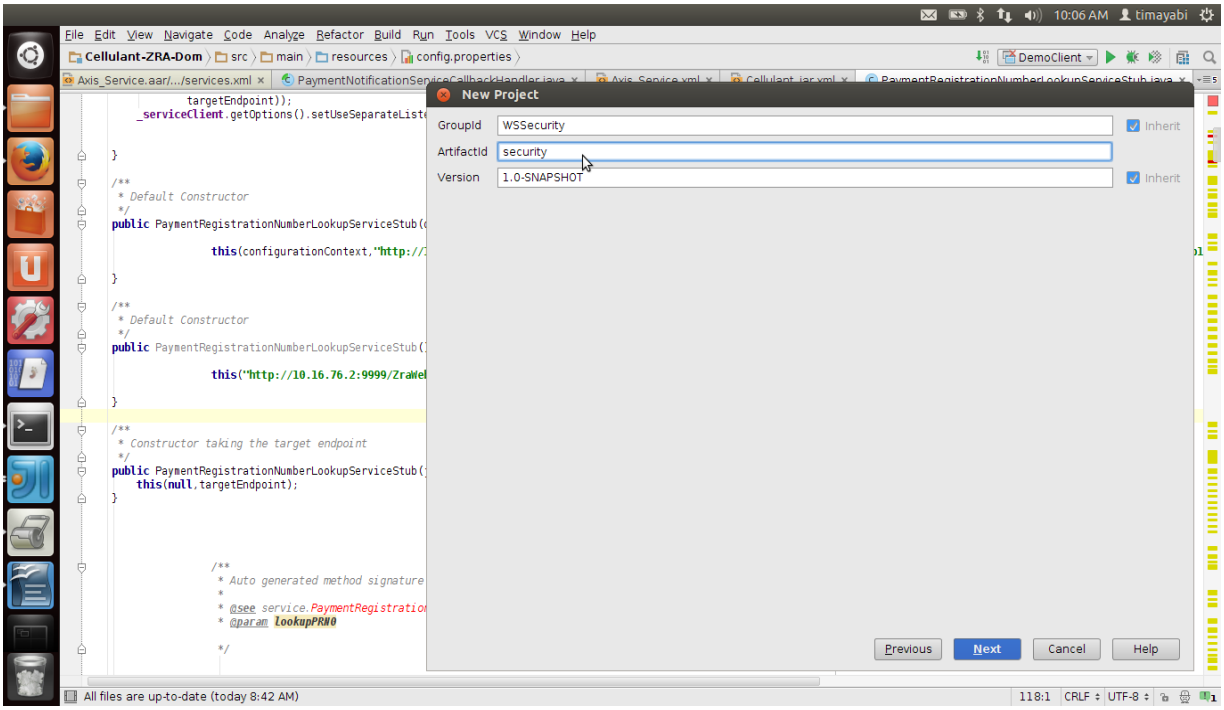
4. There are basically four services exposed to us by ZRA Domestic tax department. This include
   - PaymentRegistrationNumberLookupService
   - BankStatementNotificationService
   - PaymentNotificationService
   - EODPaymentNotificationReportService
5. Next we are going to load the wsdl for one of this services using our local port 3306 which we have port fowarded to the staging server from which the ZRA services are accessible. Therefore we expect to see the wsdl for any of the services on our local browser. See the screen shots below. Note the url entered on the browser.



6. Next we go back to our intellij IDE and generate stubs using the wsdls.
7. Open up intellij IDE and follow the screen shots below.
   - Create new maven project.



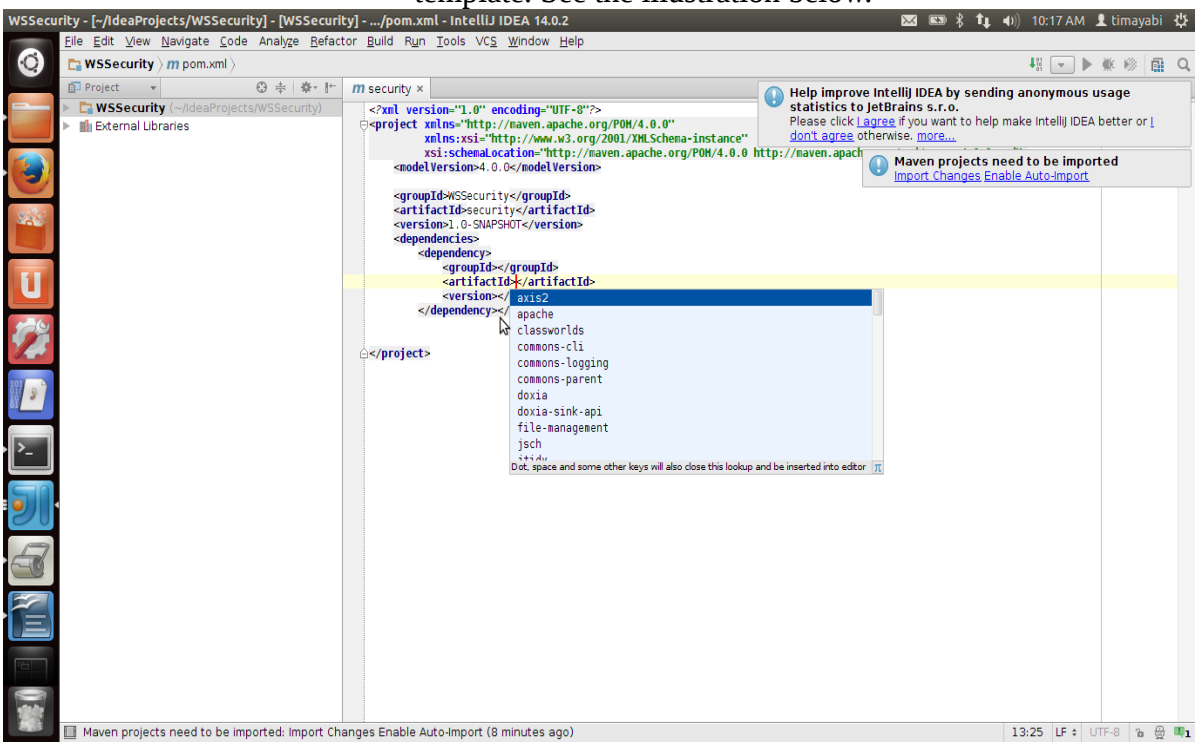
- Ignore 'create from archetype check box'
- Also ensure project SDK ponits to your current java version.
- Click next and enter group and artifact id.



- Click next and enter project name then finish
- The first page you will see is the pom.xml file(Project Object Model) file. This file provides an environment to include and define your maven dependancies.
- Before we go any further we will need to add some dependancies and this include, Axis 2 version 1.6.1, xml schema version 1.4.7,

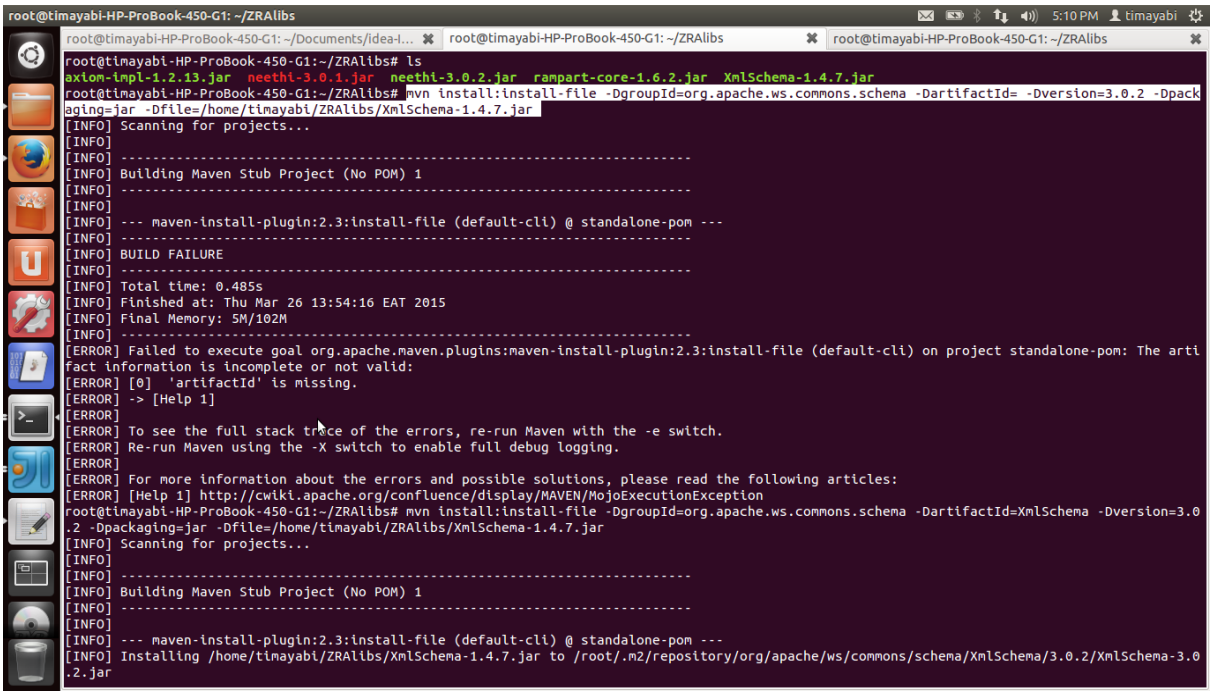
wss4j version 1.6.17, neethi version 3.0.3, axiom – impl version 1.2.14, rampart – core version 1.6.2.

– Some dependancies may be available on the IDE and to add them just go to pom.xml and press alt+insert then select dependancy template. See the illustration below.

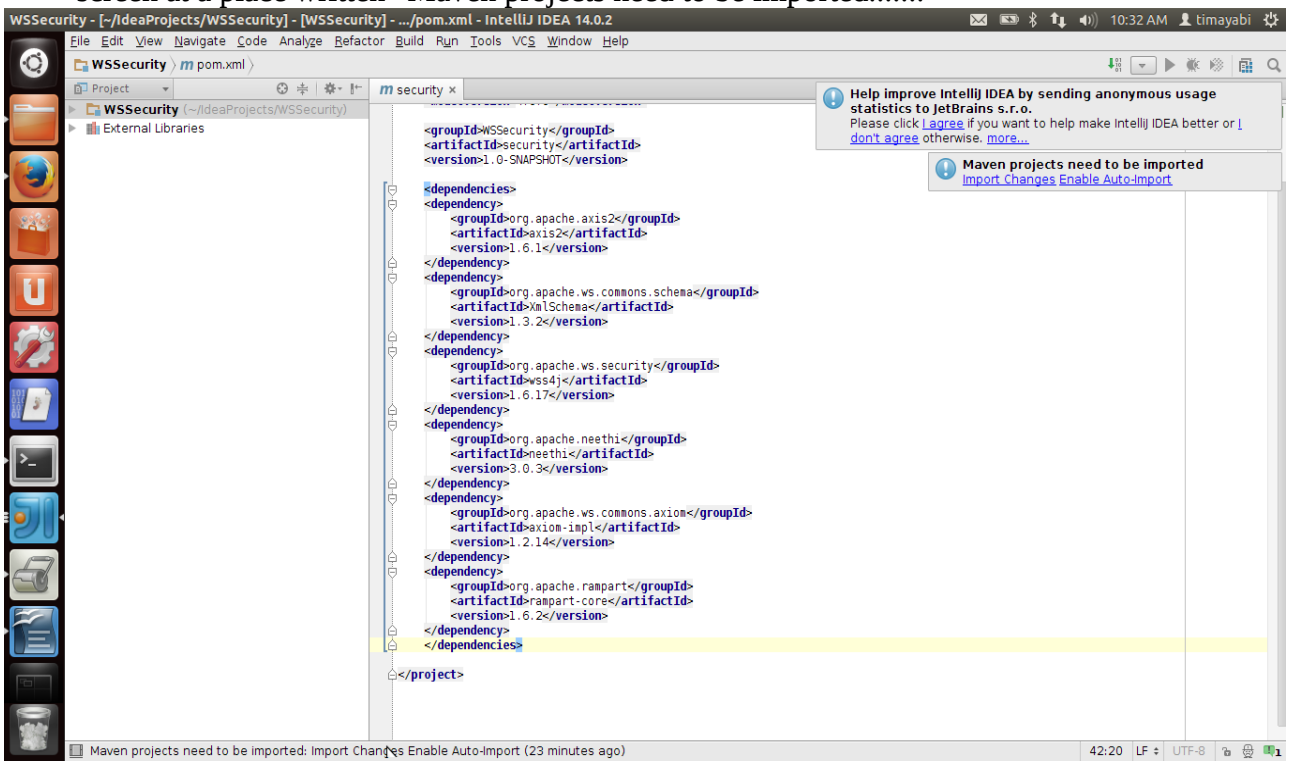


– If any of the dependancy is not available then we will have to download them and manually install into the local maven repository.

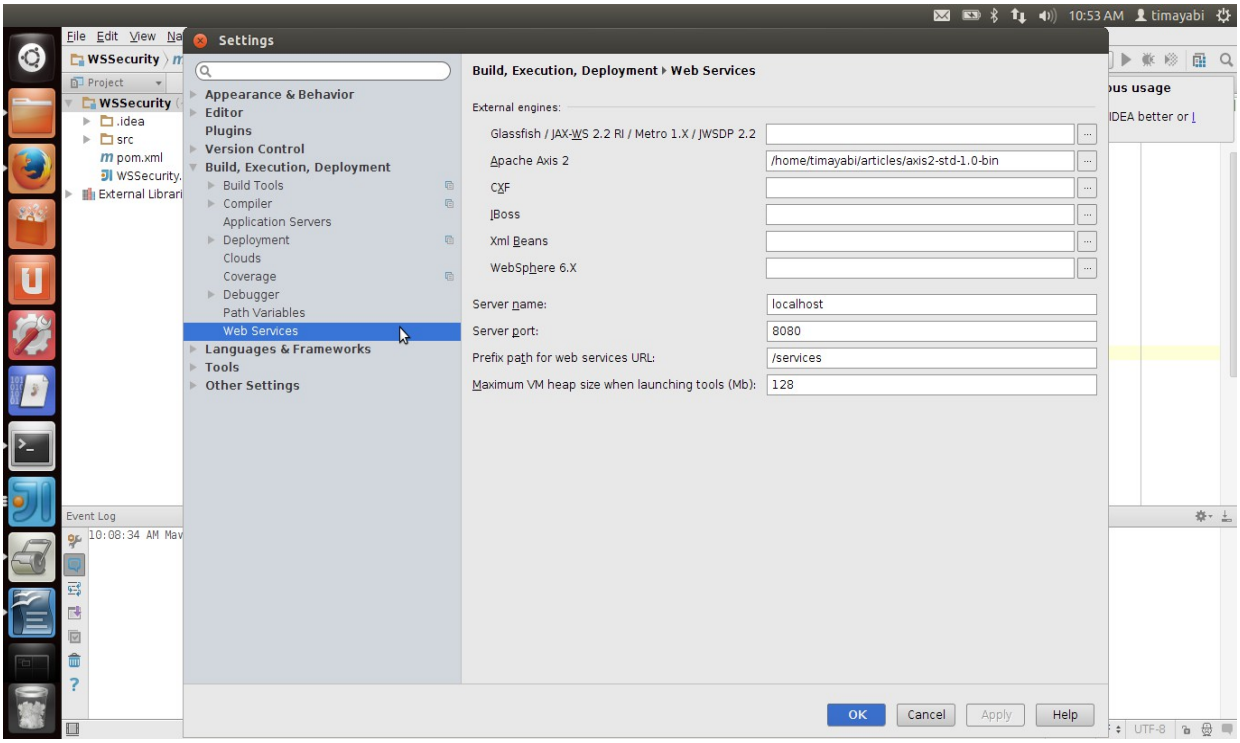
– See the screen below.



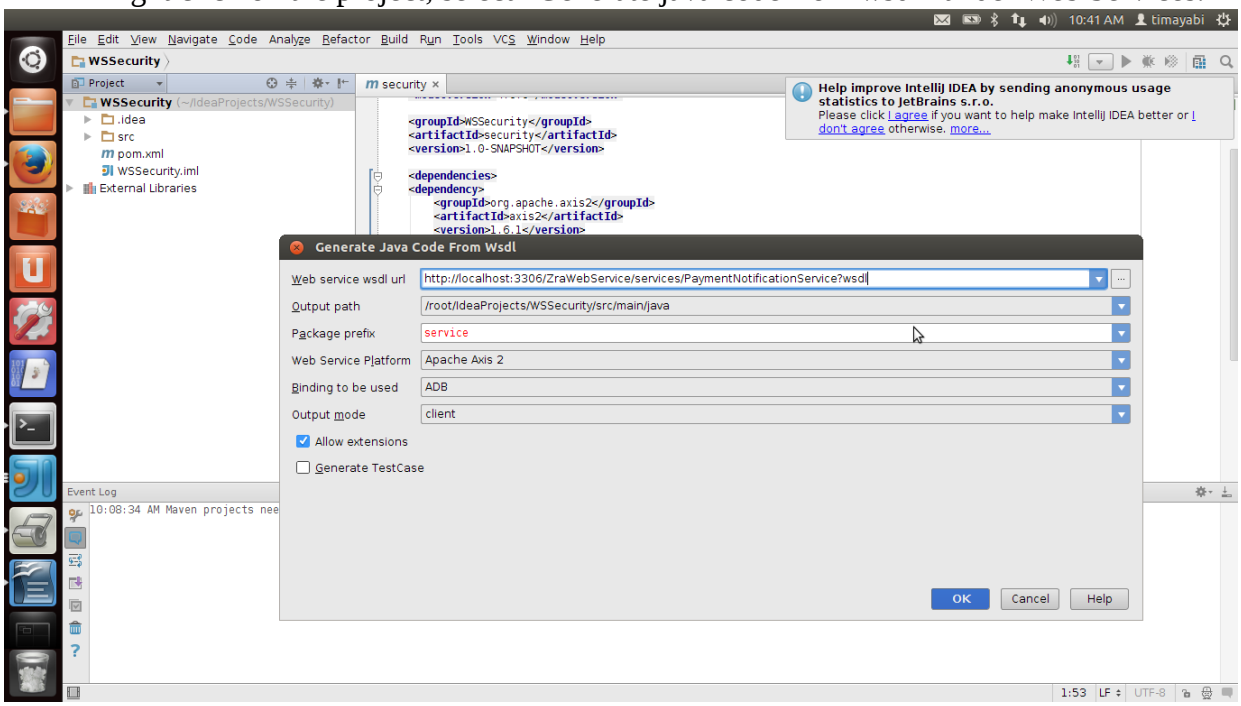
- Basically you need to open your terminal and run this command to install the downloaded maven dependancies. “mvn install:install-file -DgroupId=*org.apache.ws.commons.schema* -DartifactId=*XmlSchema* -Dversion=*3.0.2* -Dpackaging=*jar* -Dfile=/home/timayabi/ZRAlibs/XmlSchema-1.4.7.jar”
- But before you run the above command ensure that you have maven installed. If not installed just run apt-get install maven. It will do a default install of maven2.
- Go back to the IDE and add the dependancies via the pom.xml file.
- Next import the changes. See the cursor on the illustration below. It points to the bottom of the screen at a place written “Maven projects need to be imported.....”



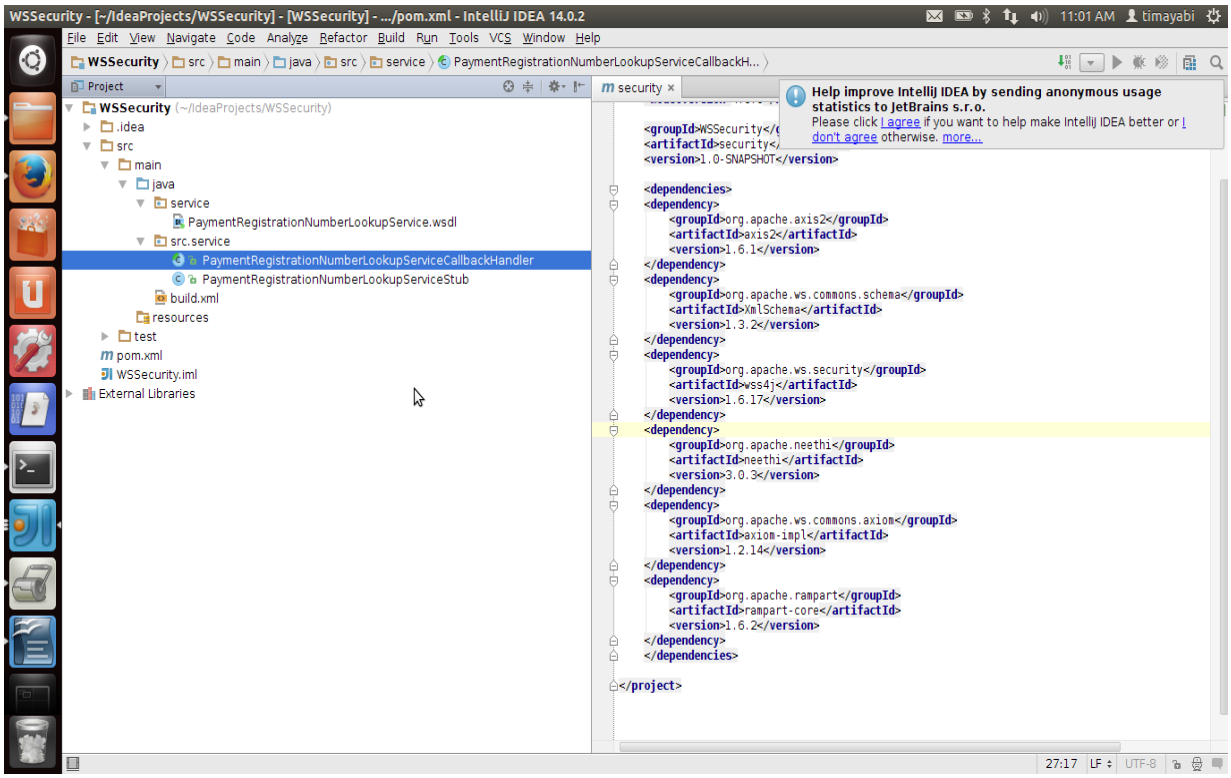
- We will also need axis2 module to generate the stubs. So download axis2-std-1.0-bin from this site and extract it to your desired folder. http://axis.apache.org/axis2/java/core/download.cgi
- Also download rampart module and extract it to your desired folder. Change directory to the rampart libraries then copy and paste the libraries to the library folder of the extracted axis module. Download the neethi library and do the same.
- Set up the axis2 path on the intellij IDE by following the screen shot below.
- On intellij, go to file ->settings, then under settings select web services. Configure the apache axis 2 path from there.



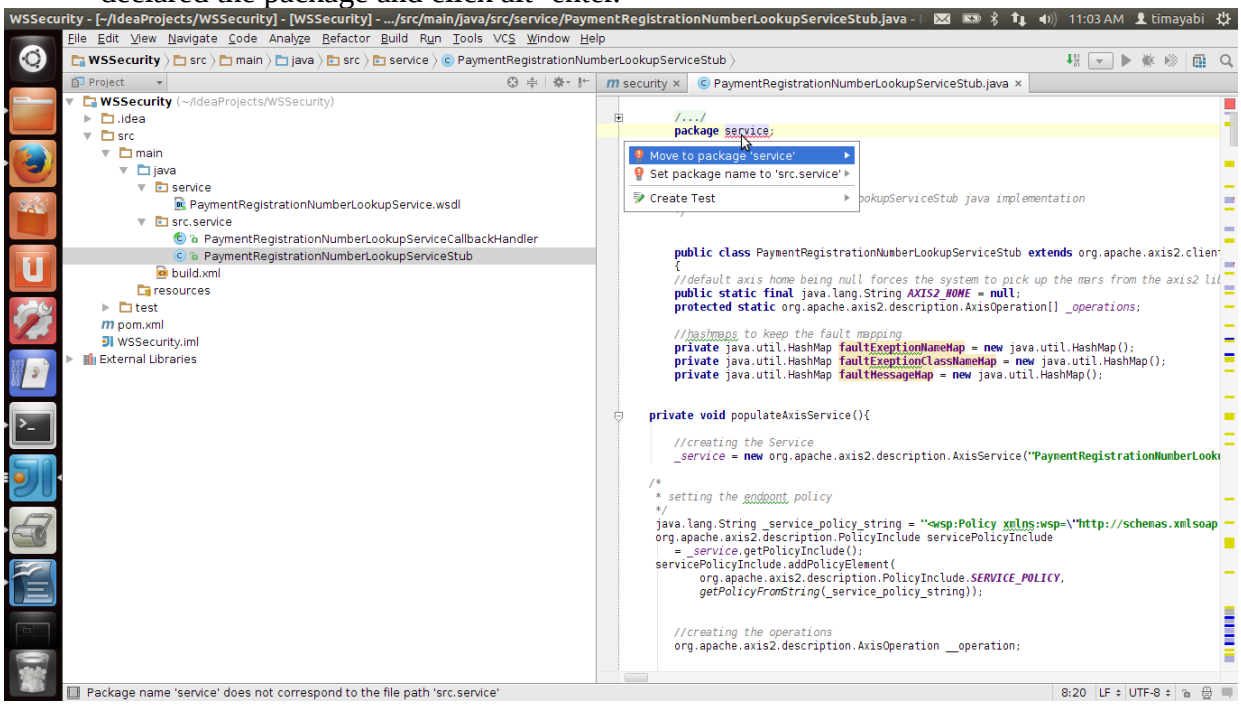
– Right click on the project, select “Generate java code from wsdl” under Web Services.



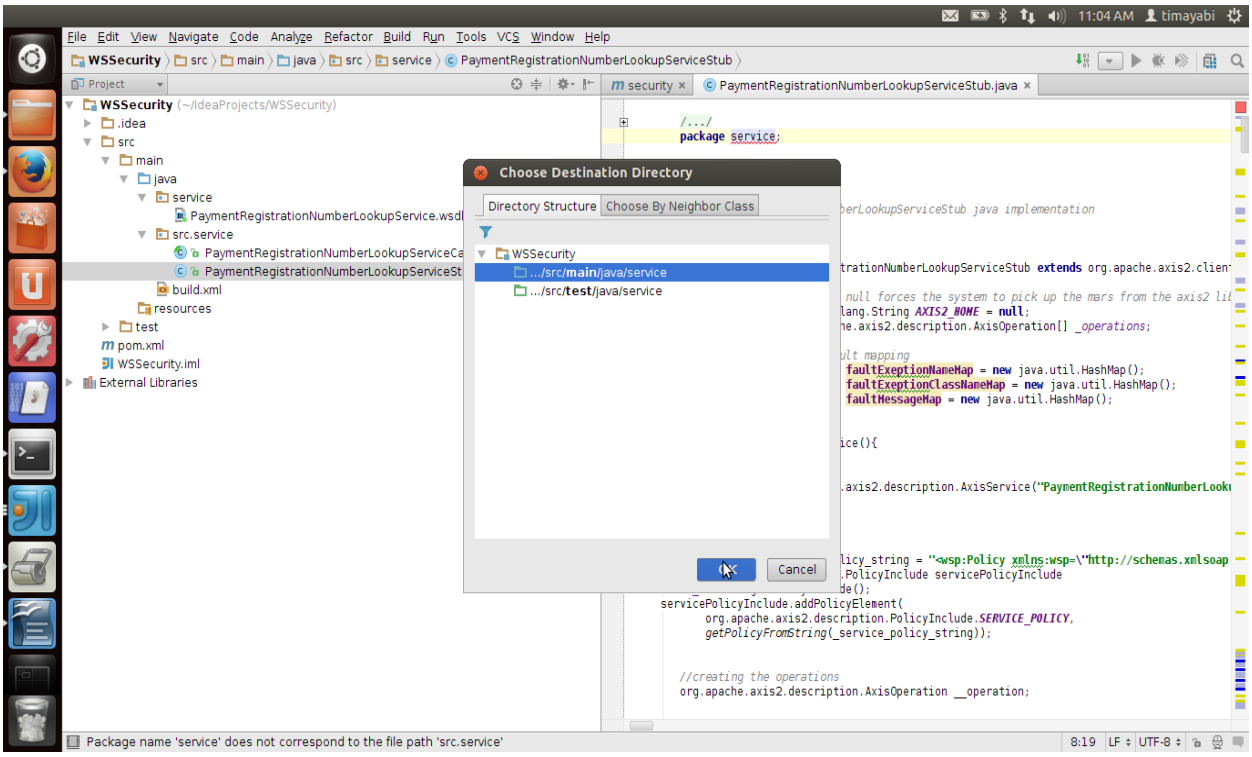
– Enter the wsdl url for any of the services , the package name and select Apache Axis 2.



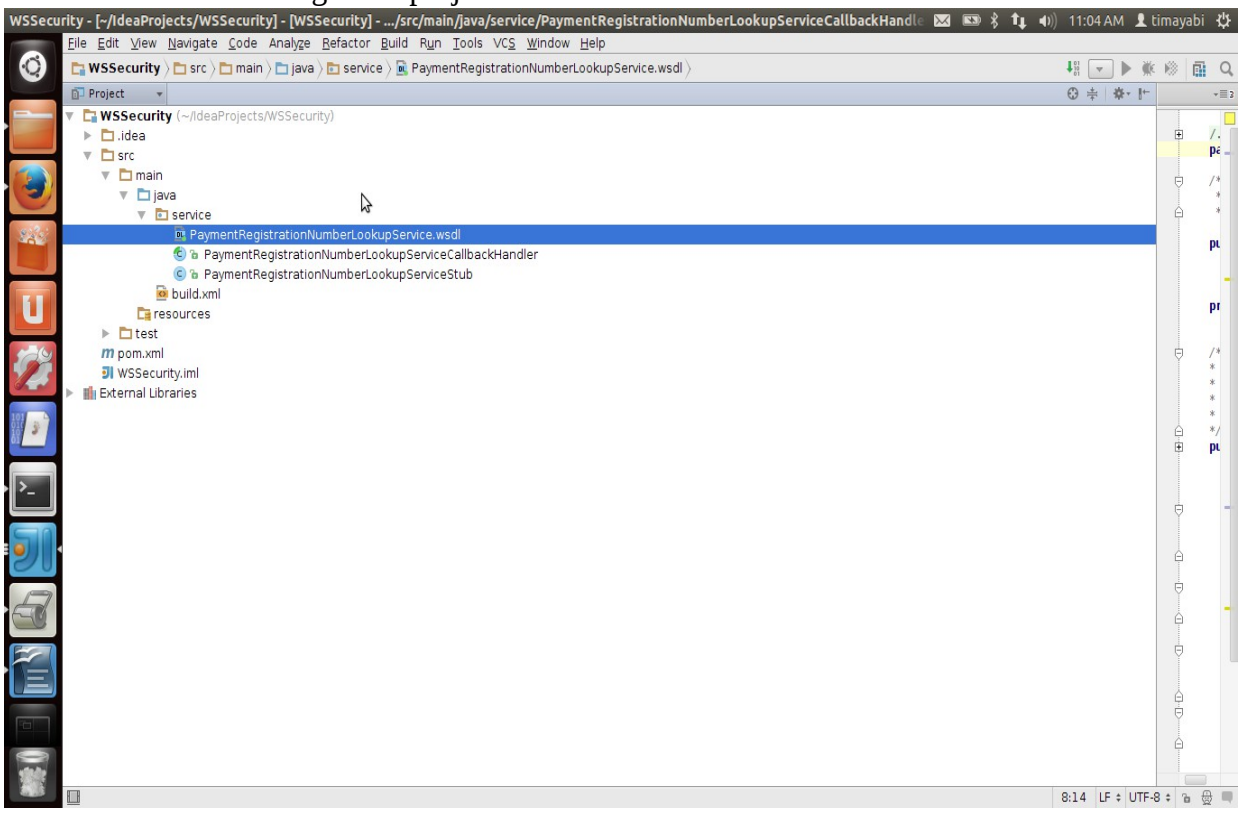
– If you click on the stub class of any of the services, you will probably get an error on line one complaining that the package cannot be found. Point the cursor to where you have declared the package and click alt+enter.



– Select the first option to move the package

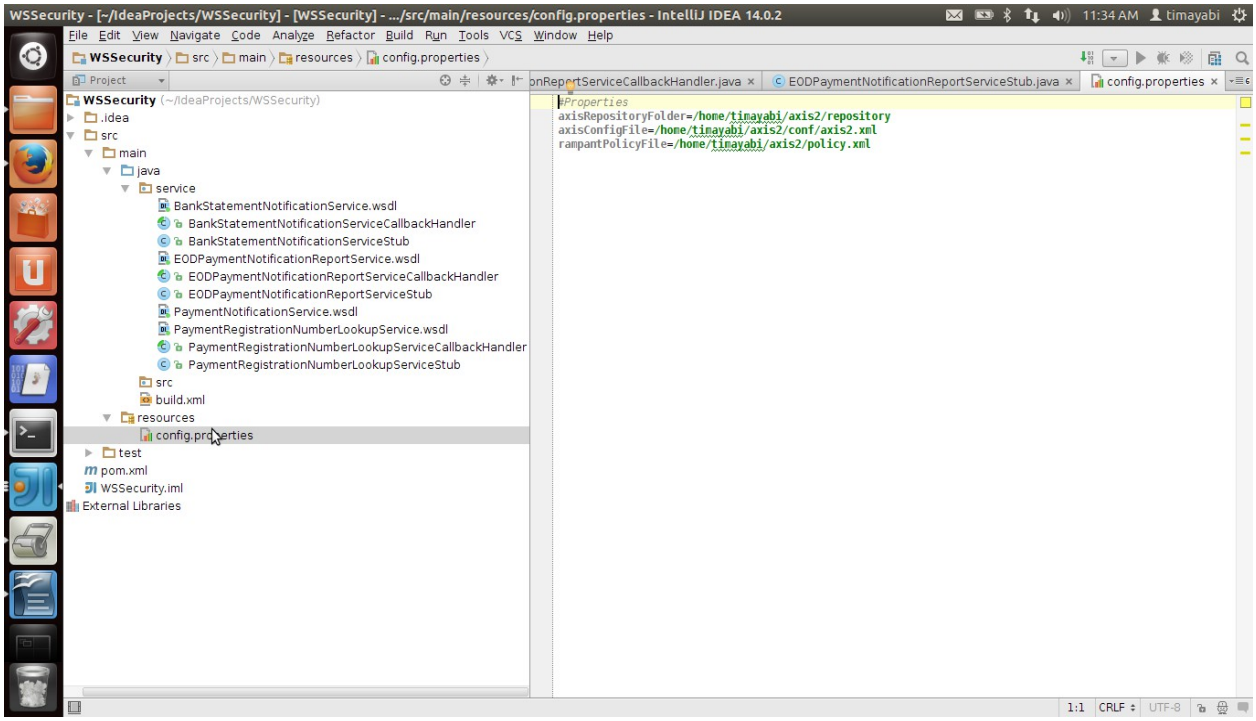


- Select the first option and click ok.
- After those changes the project tree should be the same as the illustration below.

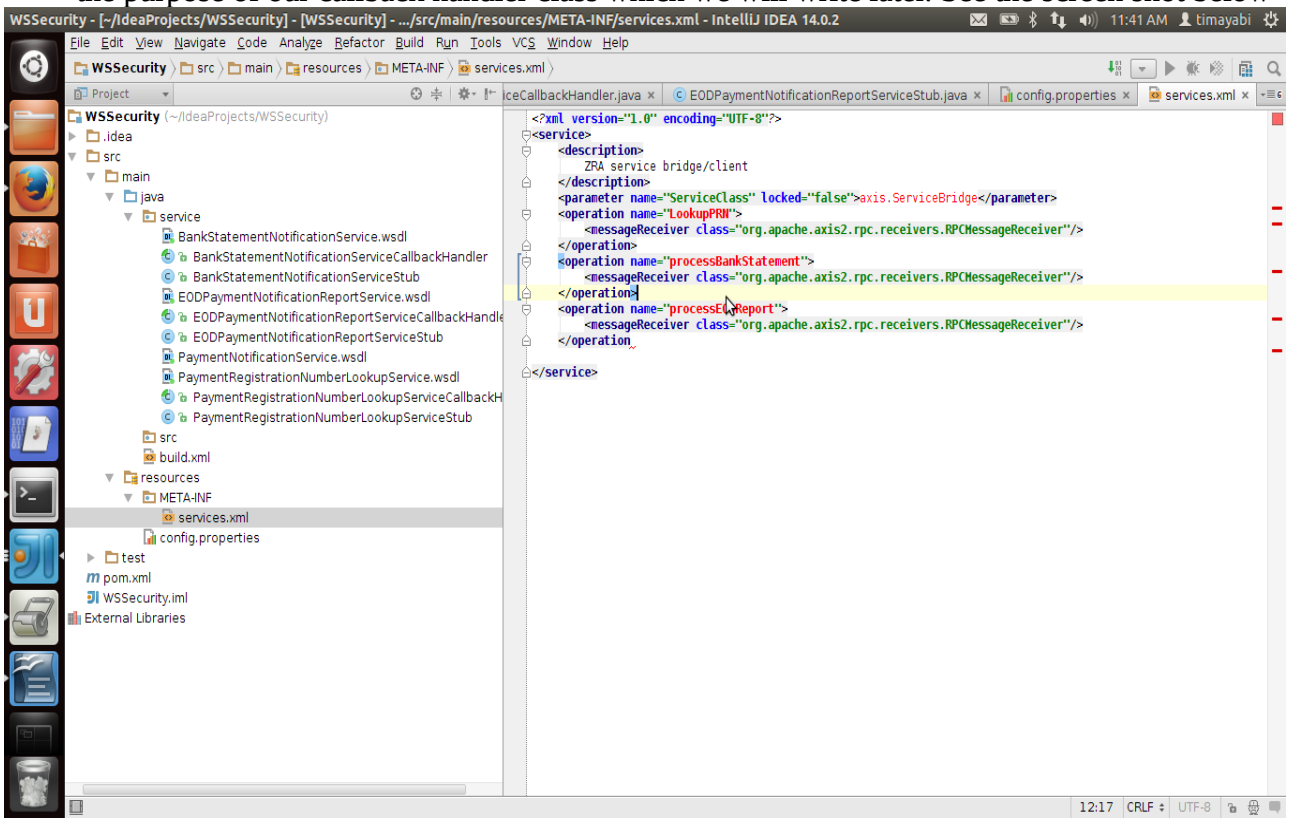


- Replicate the same procedure for all the other services.

- Next we need to have a properties file that will point to our axis repositroy, policy file and axis xml file. So under resources create “config.properties file” and add the paths as per the screen shot below.



– Still under resources create a META-INF directory where we will have a services.xml file for the purpose of our callback handler class which we will write later. See the screen shot below



Below is what you will enter on the services.xml file

```
 <?xml version="1.0" encoding="UTF-8"?>
<service>
    <description>
        ZRA service bridge/client
    </description>
    <parameter name="ServiceClass"
locked="false">axis.ServiceBridge</parameter>
    <operation name="LookupPRN">
        <messageReceiver
class="org.apache.axis2.rpc.receivers.RPCMessageReceiver"/>
    </operation>
    <operation name="processBankStatement">
        <messageReceiver
class="org.apache.axis2.rpc.receivers.RPCMessageReceiver"/>
    </operation>
    <operation name="notifyPayment">
        <messageReceiver
class="org.apache.axis2.rpc.receivers.RPCMessageReceiver"/>
    </operation>
    <operation name="processEODReport">
        <messageReceiver
class="org.apache.axis2.rpc.receivers.RPCMessageReceiver"/>
    </operation>
</service>
```

Below is what you will enter on the config.properties file.

```
#Properties
axisRepositoryFolder=/home/timayabi/axis2/repository
axisConfigFile=/home/timayabi/axis2/conf/axis2.xml
rampantPolicyFile=/home/timayabi/axis2/policy.xml

Note.Ensure that you enter your own paths.
```

- Under src->main->java create another package and name it “axis”.
- Under axis package we write the service bridge and password call handler classes.

  Below are the classes:

PasswordCallback.java

```
package axis;

import org.apache.ws.security.WSPasswordCallback;
import javax.security.auth.callback.Callback;
import javax.security.auth.callback.CallbackHandler;
import javax.security.auth.callback.UnsupportedCallbackException;
import java.io.IOException;
public class PWCallback implements CallbackHandler {
    /** Field key */
    private static final byte[] key = {
            (byte) 0x31, (byte) 0xfd, (byte) 0xcb, (byte) 0xda, (byte)
0xfb,
            (byte) 0xcd, (byte) 0x6b, (byte) 0xa8, (byte) 0xe6, (byte)
0x19,
            (byte) 0xa7, (byte) 0xbf, (byte) 0x51, (byte) 0xf7, (byte)
0xc7,
            (byte) 0x3e, (byte) 0x80, (byte) 0xae, (byte) 0x98, (byte)
0x51,
            (byte) 0xc8, (byte) 0x51, (byte) 0x34, (byte) 0x04,
    };
    /**
     * Method handle
     * Author Tim Mayabi
     * timothy.wamalwa@cellulant.com
     * @param callbacks
     * @throws IOException
     * @throws UnsupportedCallbackException
     */
    public void handle(Callback[] callbacks)
            throws IOException, UnsupportedCallbackException {
        for (int i = 0; i < callbacks.length; i++) {
            if (callbacks[i] instanceof WSPasswordCallback) {
                WSPasswordCallback pc = (WSPasswordCallback)
callbacks[i];
                //@todo: Improve logic by identifier & usage
                //System.out.println("xxx retrieve password for :
```

```
"+pc.getIdentifier() );
                if (pc.getUsage() == WSPasswordCallback.KEY_NAME) {
                    pc.setKey(key);
                } else if(pc.getIdentifier().equals("zra") ||
pc.getUsage() == WSPasswordCallback.DECRYPT) {
                    pc.setPassword("test123");//Used when encrypting
                } else if(pc.getIdentifier().equals("03404_barclays")
|| pc.getIdentifier().equals("cellulant")) {
                    pc.setPassword("test123");//Used when signing
                } else if (pc.getUsage() ==
WSPasswordCallback.SIGNATURE ) {
                    if ("cellulant".equals(pc.getIdentifier()) ||
pc.getIdentifier().equals("03404_barclays")) {
                        pc.setPassword("test123");//Used when signing
but already catered for in policy. needed?
                    }
                }
            } else {
                throw new UnsupportedCallbackException(callbacks[i],
                        "Unrecognized Callback");
            }
        }
    }
}
```

Note: *The username and password are as per what we had configured on our keystores and defined on the policy.xml file*

Below is the ***policy.xml file.***

```
<?xml version="1.0" encoding="UTF-8"?>

<!--
 !
 ! Copyright 2006 The Apache Software Foundation.
 !
 ! Licensed under the Apache License, Version 2.0 (the "License");
 ! you may not use this file except in compliance with the License.
 ! You may obtain a copy of the License at
 !
 !      http://www.apache.org/licenses/LICENSE-2.0
 !
 ! Unless required by applicable law or agreed to in writing, software
 ! distributed under the License is distributed on an "AS IS" BASIS,
 ! WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
 ! See the License for the specific language governing permissions and
 ! limitations under the License.
 !-->
<wsp:Policy wsu:Id="SigEncr"
            xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-wssecurity-
utility-1.0.xsd"
            xmlns:wsp="http://schemas.xmlsoap.org/ws/2004/09/policy">
    <wsp:ExactlyOne>
        <wsp:All>
```

```
            <sp:AsymmetricBinding
xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
                <wsp:Policy>
                    <sp:InitiatorToken>
                        <wsp:Policy>
                            <sp:X509Token
sp:IncludeToken="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/IncludeToken/AlwaysToR
ecipient">
                                <wsp:Policy>
                                    <sp:RequireThumbprintReference/>
                                    <sp:WssX509V3Token10/>
                                </wsp:Policy>
                            </sp:X509Token>
                        </wsp:Policy>
                    </sp:InitiatorToken>
                    <sp:RecipientToken>
                        <wsp:Policy>
                            <sp:X509Token
sp:IncludeToken="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy/IncludeToken/Never">
                                <wsp:Policy>
                                    <sp:RequireThumbprintReference/>
                                    <sp:WssX509V3Token10/>
                                </wsp:Policy>
                            </sp:X509Token>
                        </wsp:Policy>
                    </sp:RecipientToken>
                    <sp:AlgorithmSuite>
                        <wsp:Policy>
                            <sp:TripleDesRsa15/>
                        </wsp:Policy>
                    </sp:AlgorithmSuite>
                    <sp:Layout>
                        <wsp:Policy>
                            <sp:Strict/>
                        </wsp:Policy>
                    </sp:Layout>
                    <sp:OnlySignEntireHeadersAndBody/>
                </wsp:Policy>
            </sp:AsymmetricBinding>
            <sp:Wss10 xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
                <wsp:Policy>
                    <sp:MustSupportRefKeyIdentifier/>
                    <sp:MustSupportRefIssuerSerial/>
                </wsp:Policy>
            </sp:Wss10>
            <sp:SignedParts xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
                <sp:Body/>
            </sp:SignedParts>
            <sp:EncryptedParts
xmlns:sp="http://schemas.xmlsoap.org/ws/2005/07/securitypolicy">
                <sp:Body/>
            </sp:EncryptedParts>
            <ramp:RampartConfig xmlns:ramp="http://ws.apache.org/rampart/policy">
                <ramp:user>zra</ramp:user>
                <ramp:encryptionUser>zra</ramp:encryptionUser>
                <ramp:userCertAlias>03404_barclays</ramp:userCertAlias>
                <ramp:passwordCallbackClass>axis.PWCallback</ramp:passwordCallbackClass>
                <ramp:signatureCrypto>
                    <ramp:crypto provider="org.apache.ws.security.components.crypto.Merlin">
                        <ramp:property
name="org.apache.ws.security.crypto.merlin.keystore.type">JKS</ramp:property>
                        <ramp:property
```

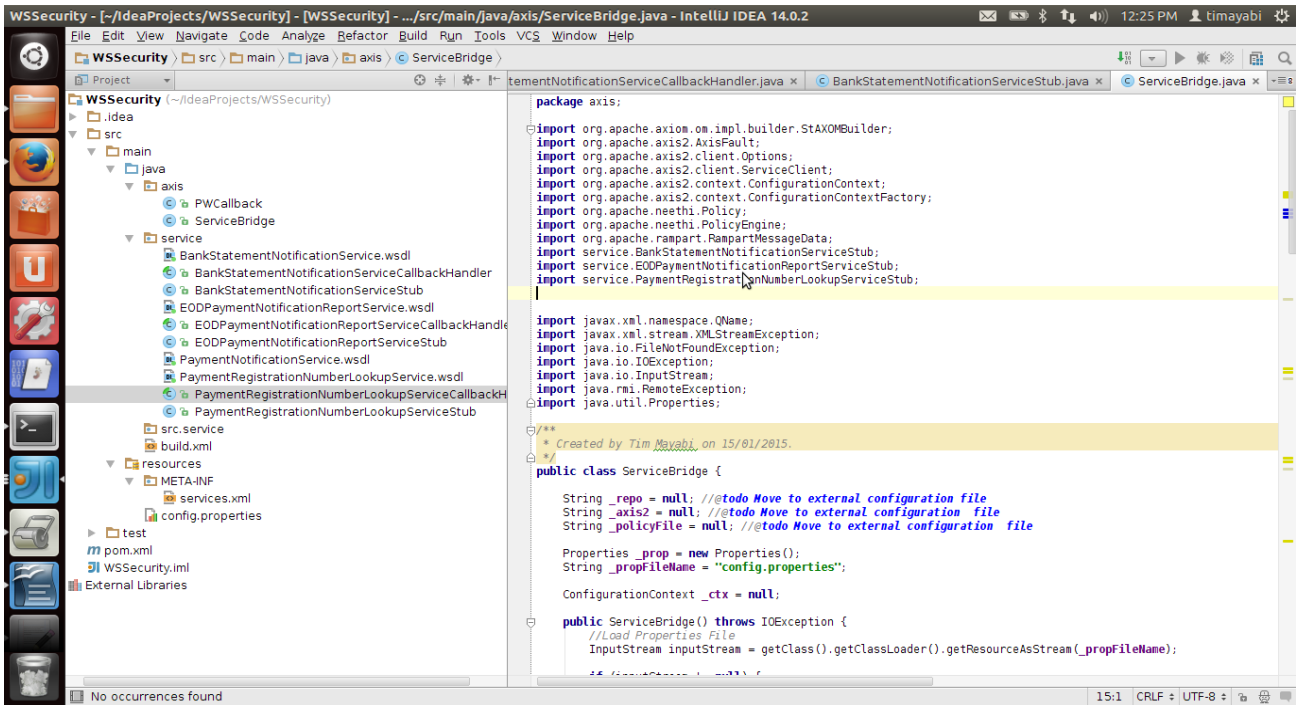
```
name="org.apache.ws.security.crypto.merlin.file">/srv/applications/ZraResources/cellulant.jk
s</ramp:property>
                        <ramp:property
name="org.apache.ws.security.crypto.merlin.alias">03404_barclays</ramp:property>
                        <ramp:property
name="org.apache.ws.security.crypto.merlin.keystore.password">test123</ramp:property>
                    </ramp:crypto>
                </ramp:signatureCrypto>
                <ramp:encryptionCypto>
                    <ramp:crypto provider="org.apache.ws.security.components.crypto.Merlin">
                        <ramp:property
name="org.apache.ws.security.crypto.merlin.keystore.type">JKS</ramp:property>
                        <ramp:property
name="org.apache.ws.security.crypto.merlin.file">/srv/applications/ZraResources/</ramp:prope
rty>
                        <ramp:property
name="org.apache.ws.security.crypto.merlin.alias">zra</ramp:property>
                        <ramp:property
name="org.apache.ws.security.crypto.merlin.keystore.password">test123</ramp:property>
                    </ramp:crypto>
                </ramp:encryptionCypto>
            </ramp:RampartConfig>
        </wsp:All>
    </wsp:ExactlyOne>
</wsp:Policy>
```

Below is the service bridge class.

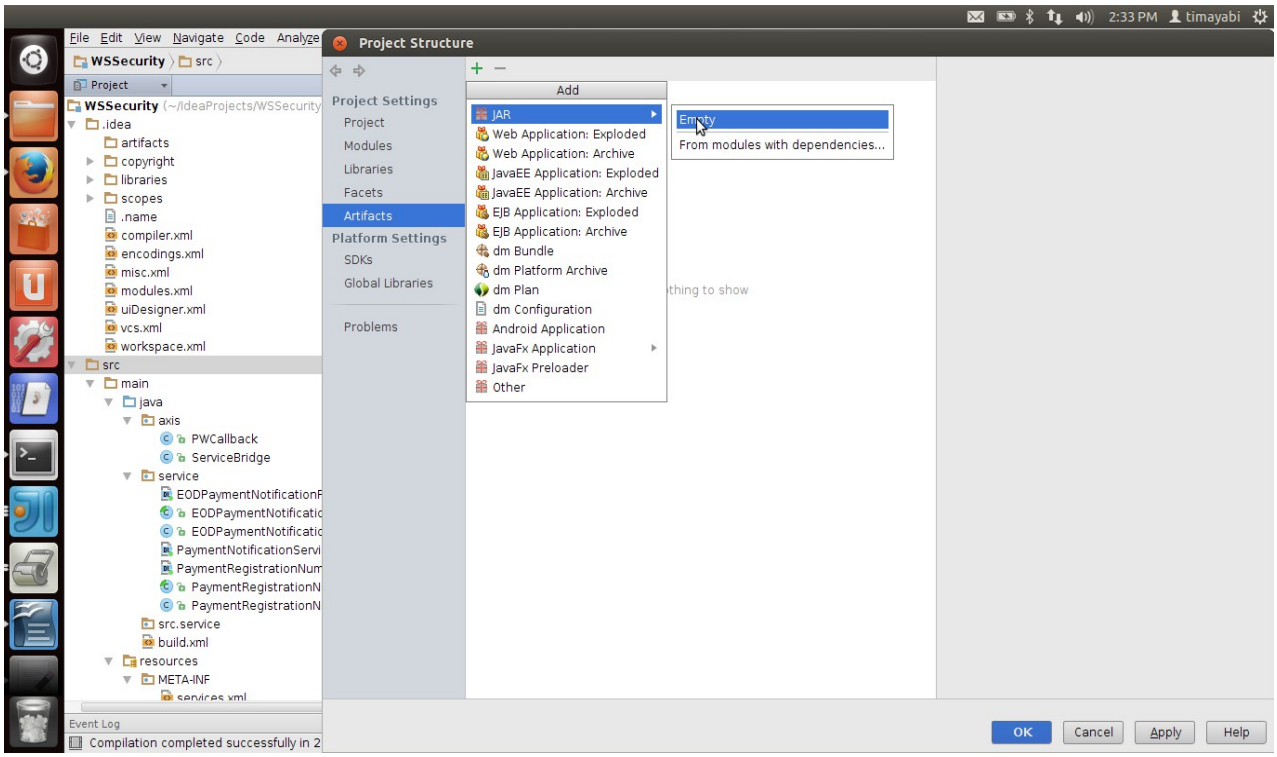
```
package axis;

import org.apache.axiom.om.impl.builder.StAXOMBuilder;
import org.apache.axis2.AxisFault;
import org.apache.axis2.client.Options;
import org.apache.axis2.client.ServiceClient;
import org.apache.axis2.context.ConfigurationContext;
import org.apache.axis2.context.ConfigurationContextFactory;
import org.apache.neethi.Policy;
import org.apache.neethi.PolicyEngine;
import org.apache.rampart.RampartMessageData;
import service.BankStatementNotificationServiceStub;
import service.EODPaymentNotificationReportServiceStub;
import service.PaymentRegistrationNumberLookupServiceStub;
import javax.xml.namespace.QName;
import javax.xml.stream.XMLStreamException;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.rmi.RemoteException;
import java.util.Properties;
/**
 * Created by Tim Mayabi on 15/01/2015.
 */
public class ServiceBridge {
    String _repo = null; //@todo Move to external configuration file
    String _axis2 = null; //@todo Move to external configuration  file
    String _policyFile = null; //@todo Move to external configuration  file
    Properties _prop = new Properties();
    String _propFileName = "config.properties";
    ConfigurationContext _ctx = null;
    public ServiceBridge() throws IOException {
        //Load Properties File
        InputStream inputStream = getClass().getClassLoader().getResourceAsStream(_propFileName);
        if (inputStream != null) {
```

```
            _prop.load(inputStream);
        } else {
            throw new FileNotFoundException("property file '" + _propFileName + "' not found in the classpath");
        }
        //Assign core properties
        _repo = _prop.getProperty("axisRepositoryFolder");
        _axis2 = _prop.getProperty("axisConfigFile");
        _policyFile = _prop.getProperty("rampantPolicyFile");
        _ctx = ConfigurationContextFactory.createConfigurationContextFromFileSystem(_repo, _axis2);
    }
    public PaymentRegistrationNumberLookupServiceStub.LookupPRNResponse
LookupPRN(PaymentRegistrationNumberLookupServiceStub.LookupPRN PRNDetails)
    {
        service.PaymentRegistrationNumberLookupServiceStub.LookupPRNResponse details = null;
        try {
            PaymentRegistrationNumberLookupServiceStub stub = new PaymentRegistrationNumberLookupServiceStub(_ctx);
            _attachPolicy(stub);//
            details = stub.lookupPRN(PRNDetails); //Call ZRA
        } catch (AxisFault axisFault) {
            axisFault.printStackTrace();
        } catch (RemoteException e) {
            e.printStackTrace();
        }
        return details;
    }
    /**
     * Send electronic bank statement for ZRA domestic tax account to ZRA
     *
     * @param statement
     * @return
     */
    public  service.BankStatementNotificationServiceStub.ProcessConsolidatedStatementResponse
processBankStatement(service.BankStatementNotificationServiceStub.ProcessConsolidatedStatement statement)
    {
        service.BankStatementNotificationServiceStub.ProcessConsolidatedStatementResponse resp = null;
        try
        {
            BankStatementNotificationServiceStub stub = new BankStatementNotificationServiceStub(_ctx);
            _attachPolicy(stub);
            resp = stub.processConsolidatedStatement(statement);
        } catch (AxisFault axisFault) {
            axisFault.printStackTrace();
        } catch (RemoteException e) {
            e.printStackTrace();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return resp;
    }
    /**
     * Send end of day report to ZRA
     * Contains all transactions that occurred the previous day
     *
     * @param report
     * @return
     */
    public  service.EODPaymentNotificationReportServiceStub.ProcessPaymentNotificationReportResponse
processEODReport(service.EODPaymentNotificationReportServiceStub.ProcessPaymentNotificationReport report)
    {
        service.EODPaymentNotificationReportServiceStub.ProcessPaymentNotificationReportResponse resp = null;
        try
        {
            EODPaymentNotificationReportServiceStub stub = new EODPaymentNotificationReportServiceStub(_ctx);
            _attachPolicy(stub);
            resp = stub.processPaymentNotificationReport(report);
```

```
        } catch (AxisFault axisFault) {
            axisFault.printStackTrace();
        } catch (RemoteException e) {
            e.printStackTrace();
        }
        return resp;
    }
    /**
     * Attach signature and encryption policy to the service
     * Requires Rampant module
     *
     * @param stub
     */
    private void _attachPolicy (org.apache.axis2.client.Stub stub)
    {
        try {
            ServiceClient client = stub._getServiceClient();
            Options clientOptions = client.getOptions();
            Policy _policy= loadPolicy(_policyFile);
            clientOptions.setProperty(RampartMessageData.KEY_RAMPART_POLICY, _policy);
            client.setOptions(clientOptions);
            client.engageModule(QName.valueOf("addressing"));
            client.engageModule(QName.valueOf("rampart"));
            stub._setServiceClient(client);
        } catch (AxisFault axisFault) {
            axisFault.printStackTrace();
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        } catch (XMLStreamException e) {
            e.printStackTrace();
        }
    }
    private Policy loadPolicy(String filePath)
            throws XMLStreamException, FileNotFoundException {
        StAXOMBuilder builder = new StAXOMBuilder(filePath);
        return PolicyEngine.getPolicy(builder.getDocumentElement());
    }
}
```

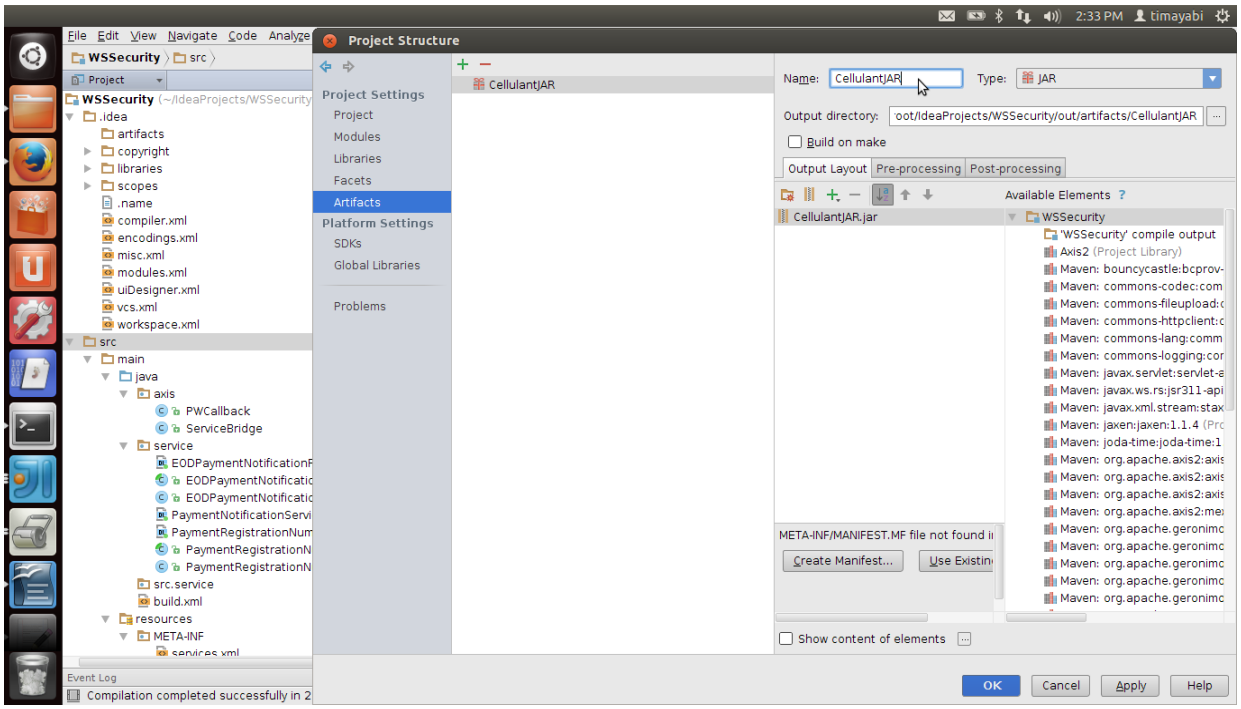


- The next step is to have our artifacts so that we have our .aar file and jar file. This files will files will be copied to the axis module configured on tomcat.
- On the project, create artifact folder under .idea directory.
- Navigate to file->project structure and select Artifacts as in the diagram below.

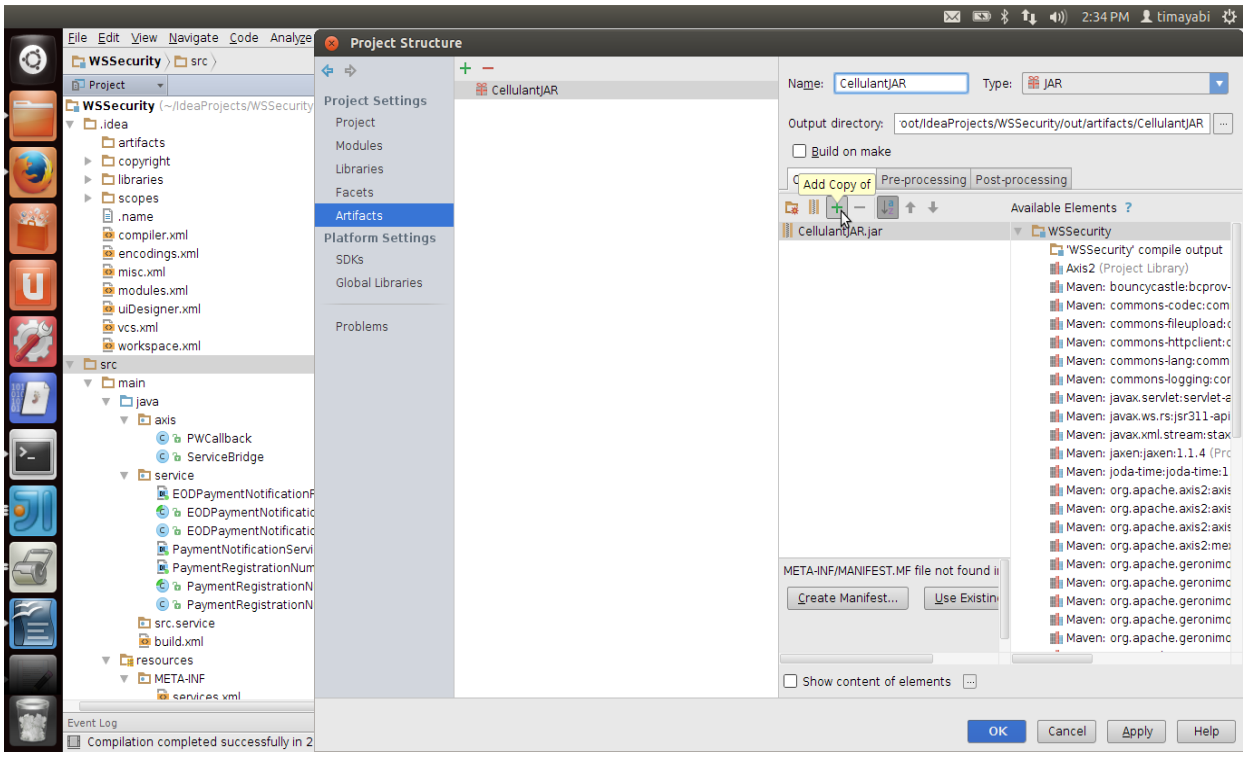


- Click on the plus sign, select JAR and then Empty to create a new artifact.

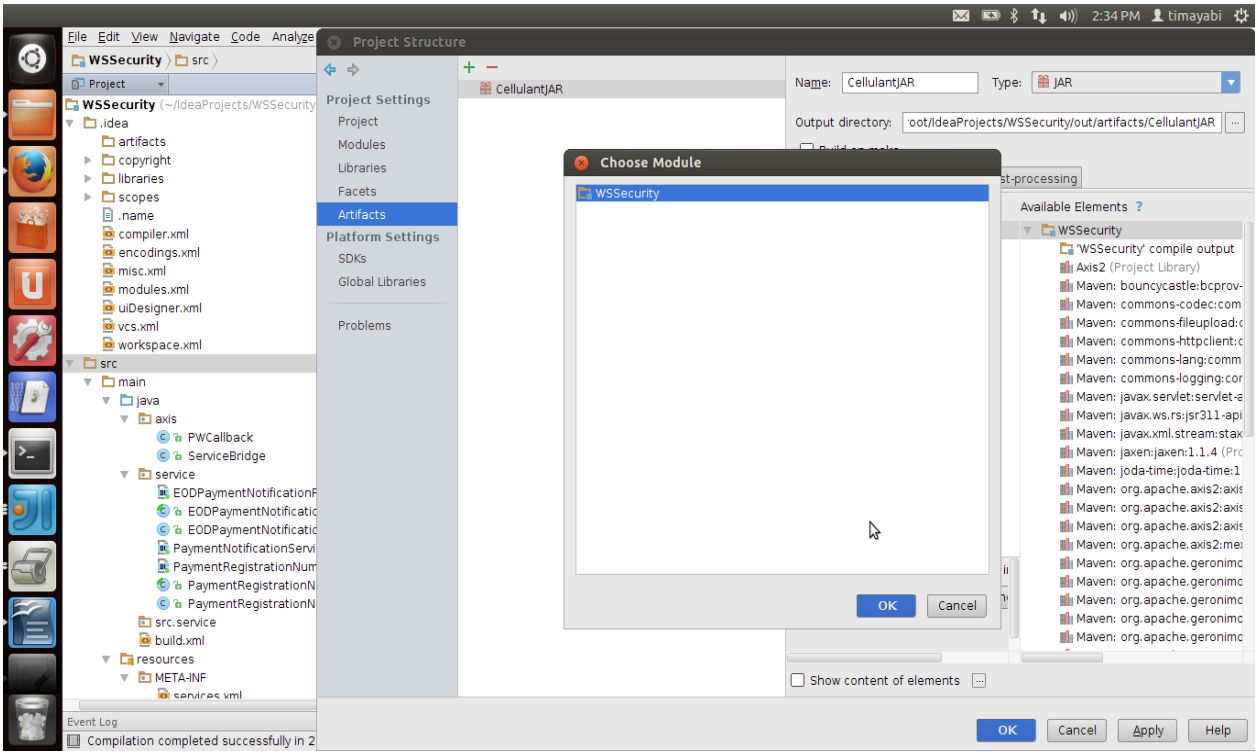
- Go to right hand side of the wizard and give your artifact a name. See the screen shot below.



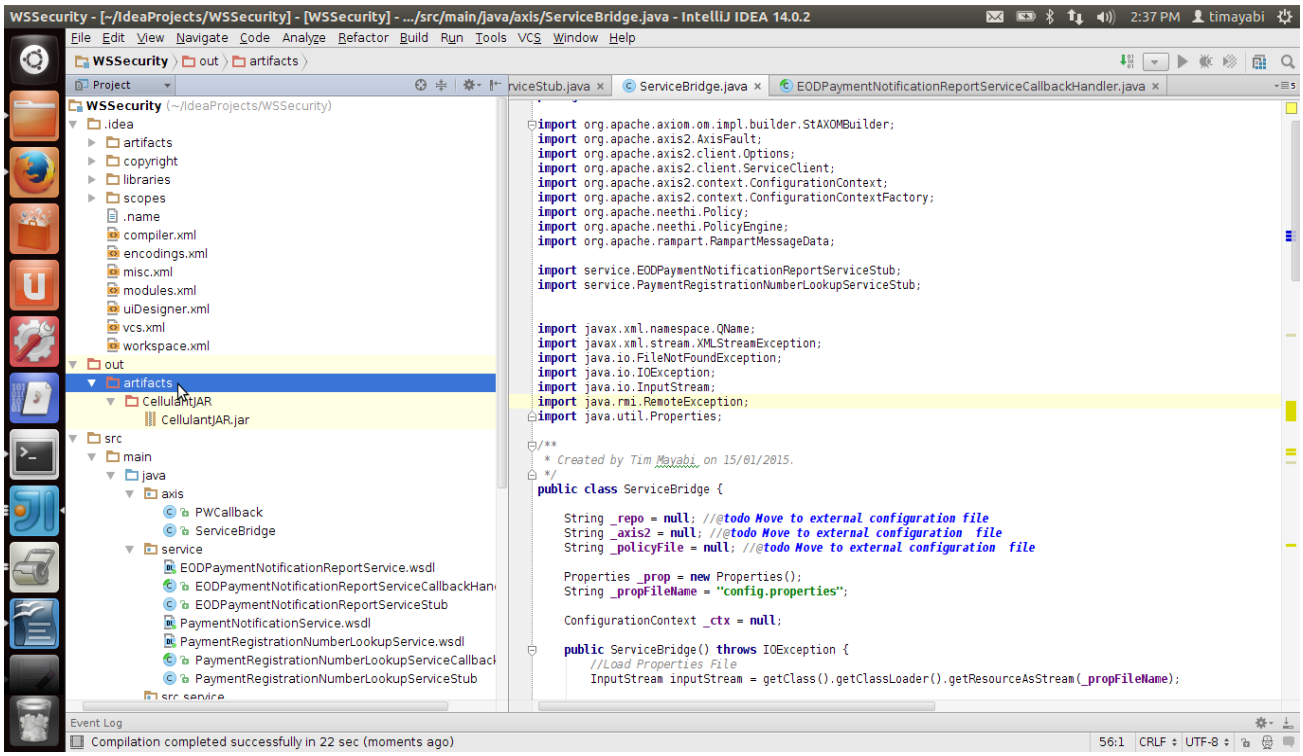
- On the output layout field, click on the plus sign so that it adds the output module onto which the artifact will be added once it is built.



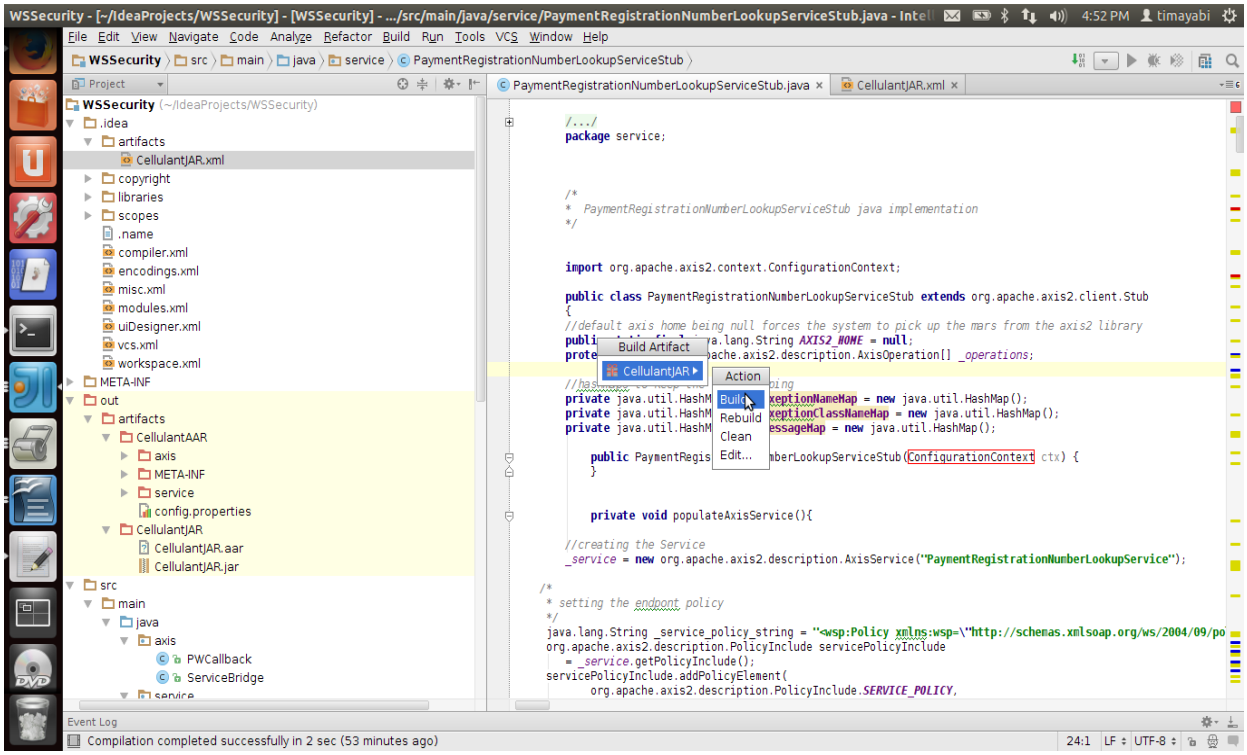
- After clicking choose a module.



- Click “ok”, “Apply” and then “ok”. Your structure should be as per the screen shot below.



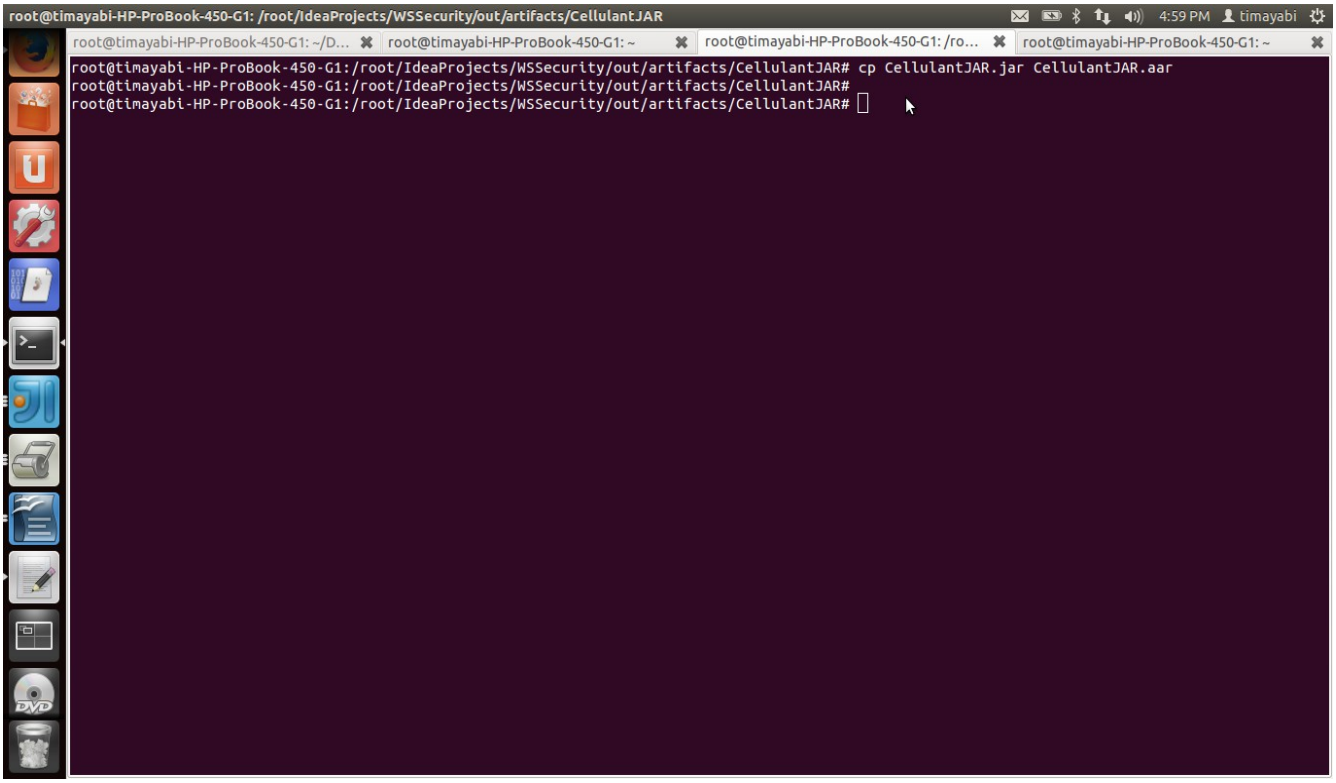
- Go to build, select artifacts and build artifact.



- Finally go to build and rebuild the whole project.

Deploying the Service.

1. Open terminal and change directory to your project folder.
2. Under out->artifacts, copy .jar file to .aar file. See the screen shot below.



3. Copy the .aar file to your axis module which you had built inside tomcat web server. On my machine this is the path.

   /usr/local/apache-tomcat-7.0.47/webapps/axis2/WEB-INF/services

4. Also copy the .jar file to the lib directory of the same path. On my machine this is the path.

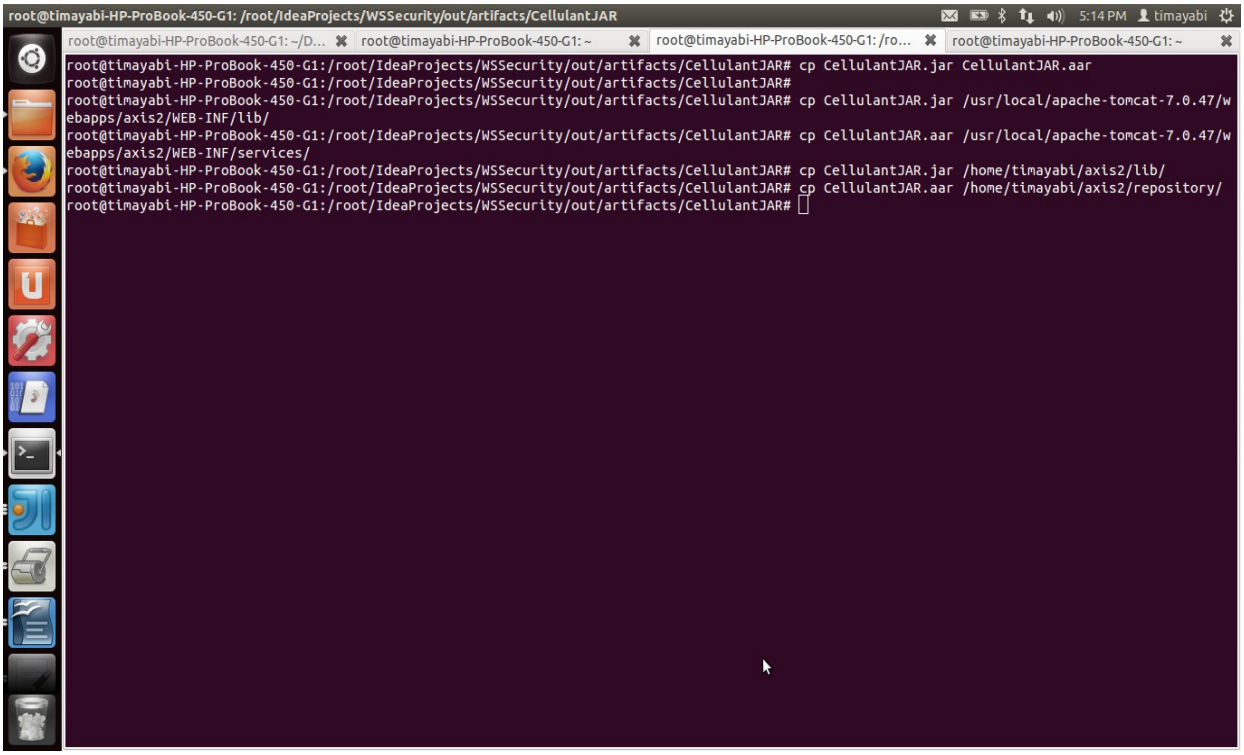
   /usr/local/apache-tomcat-7.0.47/webapps/axis2/WEB-INF/lib/

5. Again we need to copy the .aar file to the axis repository. Remember that this repository we had defined it on our services.xml file. In my case I will copy the file to

   /home/timayabi/axis2/repository

   then I will also copy the .jar file to /home/timayabi/axis2/lib

   The screen shot below shows all the steps described in this deployment procedure.

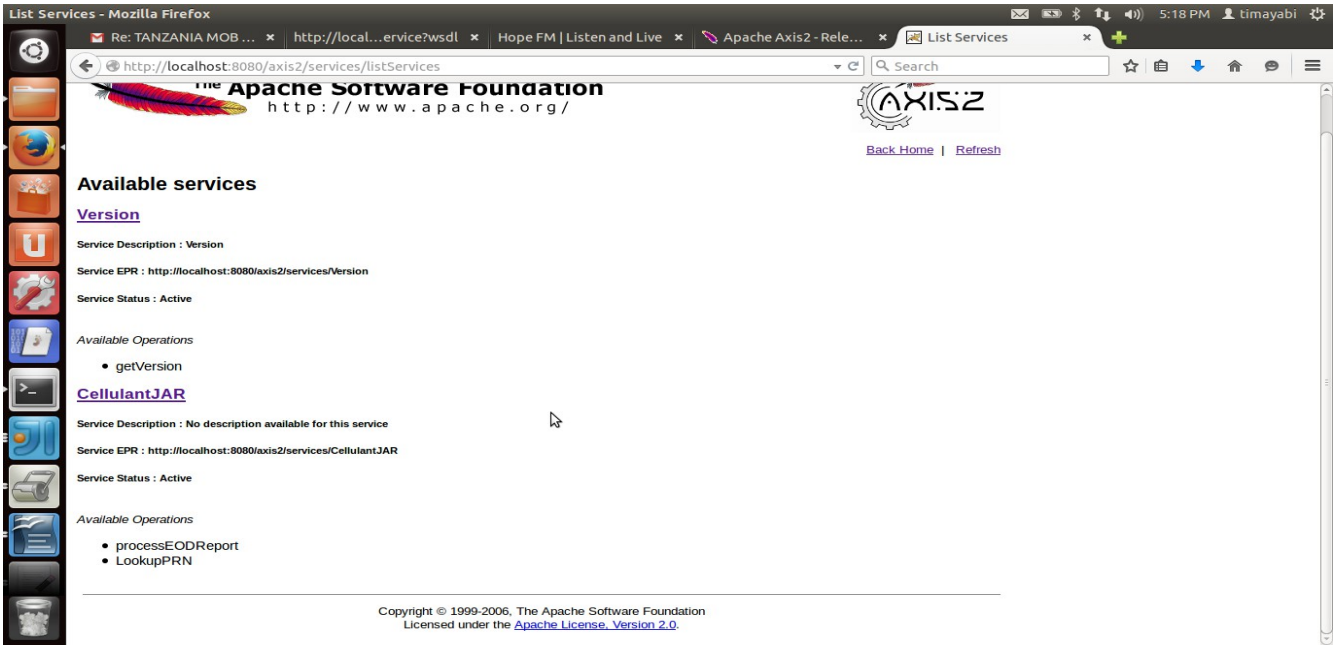


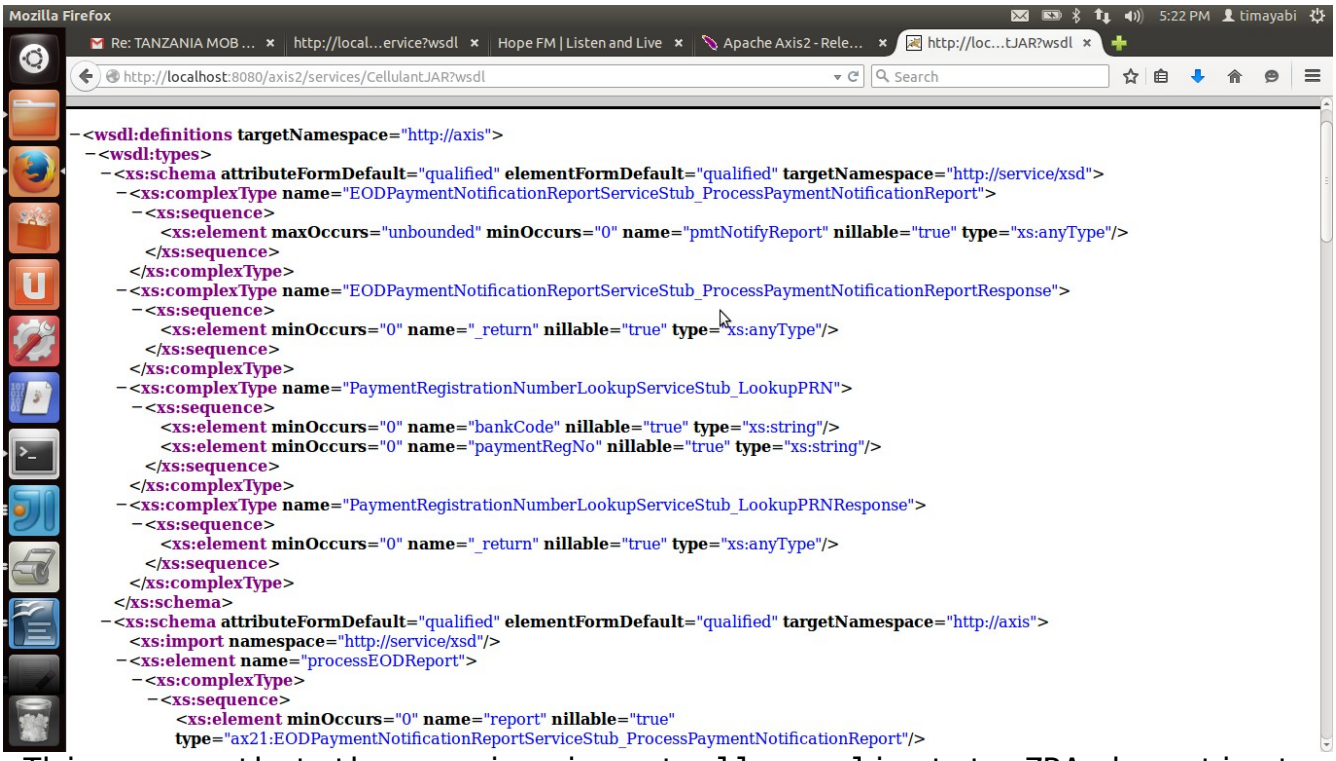
6. After deployment remember to restart tomcat.

7. Finally enter this url on the browser http://localhost:8080/axis2/services/listServices

   and expect to see the deployment result as per the screen shot below

8. When you also click on the service, in this case “CellulantJAR” you will be able to view the wsdl on your browser.

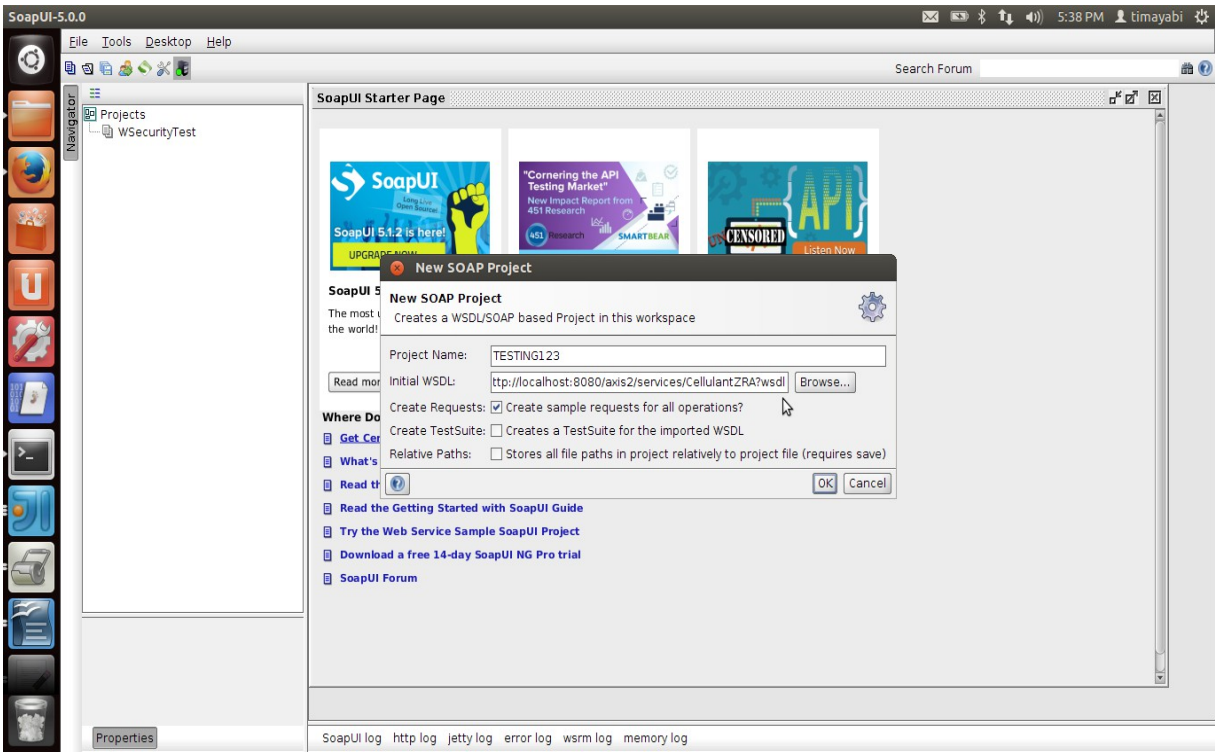




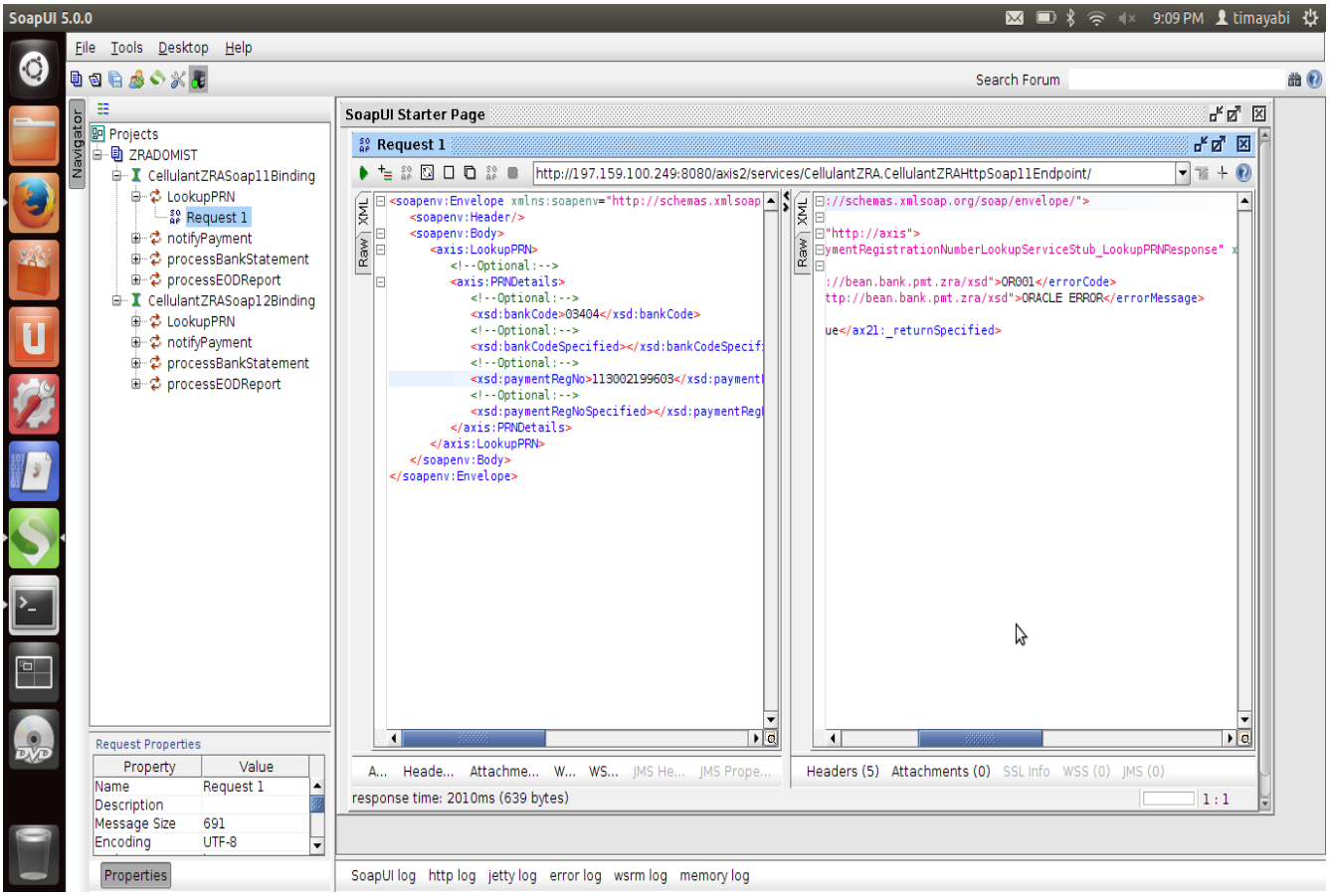
This means that the service is actually a client to ZRA domestic tax service but we will need to write a client to invoke this service so that it can relay the params to ZRA domestic tax service. The good thing is that the client can be written in any programming language you are comfortable with.

Testing the Service.

Using Soap UI follow create a new project and copy the service wsdl.



Enter some values to test the service as shown on the screenshot below.



The above screen shot shows the request and the response from ZRA.

Using a php client I created to invoke this service I still got the same response. See the log client below.

```
 2015-03-31 21:08:58 | INFO | /var/www/html-ke/zra/ZRAFetchTaxesAPI.php | fetchAssesmentPRN | LINE:455 | -1 | getPRNParams params

2015-03-31 21:08:58 | INFO | /var/www/html-ke/zra/ZRAFetchTaxesAPI.php | fetchAssesmentPRN | LINE:463 | -1 | getPRNParams params Array

(

    [bankCode] => 03404

    [paymentRegNo] => 113002199603

)



2015-03-31 21:08:59 | INFO | /var/www/html-ke/zra/ZRAFetchTaxesAPI.php | fetchAssesmentPRN | LINE:527 | -1 | My response:::::::::::::::::Array

(
```

```
    [0] => Array
        (
            [amountToBePaid] =>
            [amountPaidSpecified] =>
            [errorCode] => OR001
            [errorCodeSpecified] =>
            [errorMessage] => ORACLE ERROR
            [errorMessageSpecified] =>
            [expiryDate] =>
            [expiryDateSpecified] =>
            [paymentRegDate] =>
            [paymentDateSpecified] =>
            [paymentRegNo] =>
            [paymentRegNoSpecified] =>
            [status] =>
            [statusSpecified] =>
            [taxPayerName] =>
            [taxPayerNameSpecified] =>
            [tin] =>
            [tpin] =>
              [tinSpecified] =>
        )

)
```

This clearly indicates that the ZRA oracle server is not up and running. In such a scenario, one should liase with a technical person from ZRA.

Conclusion

The tools, procedures and instructions explained here is all that is required to integrate into any other web service that uses policy files to sign, encrypt and authenticate soap messages