**Prescription Monitoring Program  
Information Exchange Service**

**Secure Java Metro PMIX Reference Implementation**

**February 2012**

**TABLE OF CONTENTS**

[1 Introduction 3](#_Toc317071400)

[2 Prerequisites 3](#_Toc317071401)

[3 Setting up the Eclipse workspace:- 4](#_Toc317071402)

[4 Running the server 4](#_Toc317071403)

[5 Running the client 4](#_Toc317071404)

[6 To generate a self-signed certificate using Java’s ‘keytool’ utility 4](#_Toc317071405)

[6.1 Create the server keystore: 5](#_Toc317071406)

[6.2 Creating the client truststore: 5](#_Toc317071407)

# 

# Introduction

This Java reference implementation for the Prescription Monitoring Program Information Exchange service (PMIX) is built to conform to the JAX-WS standard and uses the Glassfish (Metro) implementation for Java, and implements WS-Security. It has been built using Eclipse IDE for Java EE developers. This implementation adheres to the following standards:-

* Eclipse Helios or Indigo
* **GlassFish Server Open Source Edition 3.1.1**
* Metro 2.1
* JAX WS version – 2.2
* JDK 1.6

# Prerequisites

The projects in the *PMIX Reference Implementation* folder can be imported into a workspace in Eclipse.

* The version of Eclipse being used should support the Web Tools Platform (WTP) (Eclipse Helios or Indigo). The Helios version can be found here:- <http://www.eclipse.org/downloads/download.php?file=/technology/epp/downloads/release/helios/SR2/eclipse-jee-helios-SR2-win32.zip&url=http://www.gtlib.gatech.edu/pub/eclipse/technology/epp/downloads/release/helios/SR2/eclipse-jee-helios-SR2-win32.zip&mirror_id=337>
* The Glassfish server must be installed and correctly configured in Preferences in Eclipse. The 3.1.1 release for Windows can be downloaded here:- <http://download.java.net/glassfish/3.1.1/release/glassfish-3.1.1.zip>

# Setting up the Eclipse workspace:-

1. Import all the projects into a workspace in Eclipse.

# Running the server

1. The PMPService.war file is located in the *PMIX Secure Java Metro Implementation* folder. Place this file in the ‘autodeploy’ folder in the Glassfish installation on the local machine. If using the default domain, the folder is located at \glassfish\domains\domain1\autodeploy.
2. Place the pmpKeystore.jks file in the \glassfish\domains\domain1\config folder. This jks file contains a self-signed certificate.
3. Open the domain.xml file located in the config folder. Replace all instances of ‘cacerts.jks’ and ‘keystore.jks’ with ‘pmpKeystore.jks’. Also replace all instances of the alias name ‘s1as’ with ‘certificatekey’. This ensures that the server uses the certificate provided instead of the default certificates. Save the changes to domain.xml
4. Start the server with the following command typed into a command prompt window – *asadmin start-domain domain1*.

The server can be stopped with the command – *asadmin stop-domain domain1*

1. Browse to <https://mm.test.rxcheck.org:18812/PMP_Service/pmp?wsdl> – The wsdl should be hosted here.

# Running the client

1. The PMP Client Application file is the Client project. The service endpoint address can be changed in the “address” parameter in the client.properties file.
2. Right click on ClientApplication.java in the src/client folder and *Run as a Java Application.*
3. Enter sample information and click on Provide or Receive to send a synchronous message.

Note: The provided sample uses a self-signed certificate generated using Java’s ‘keytool’ utility. To generate a new self-signed certificate, the following steps may be followed-

# To generate a self-signed certificate using Java’s ‘keytool’ utility

Browse to a new folder where the server and client keystore and truststore are to be generated

## Create the server keystore:

1. openssl req -x509 -days 3650 -newkey rsa:1024 -keyout servicekey.pem -out servicecert.pem -passout pass:changeit
2. openssl pkcs12 -export -inkey servicekey.pem -in servicecert.pem -out service.p12 -name certificatekey -passin pass:changeit -passout pass:changeit
3. keytool -importkeystore -destkeystore pmpKeystore.jks -deststorepass changeit -deststoretype jks -srckeystore service.p12 -srcstorepass changeit -srcstoretype pkcs12
4. keytool -list -keystore pmpKeystore.jks -storepass changeit -v
5. keytool -exportcert -alias certificatekey -storepass changeit -keystore pmpKeystore.jks -file service.cer
6. keytool -printcert -file service.cer
7. rm \*.pem \*.p12

Notes –

* When running command 1, enter geographic and company information. The password used here is ‘changeit’, any password can be used in its place.
* Command 2 exports a certificate with alias name ‘certificatekey’.
* Command 3 places the certificate in a Java recognized jks format. The password is again set to ‘changeit’. This password can be same as, or different from the one used in Command 1
* Command 4 displays the content of the keystore.
* Command 5 exports the certificate out of the keystore as a .cer file.
* Command 6 displays the contents of the certificate.
* Command 7 deletes the .pem and .p12 files that are no longer needed.

## Creating the client truststore:

The client truststore is created in a way similar to the keystore.

1. openssl req -x509 -days 3650 -newkey rsa:1024 -keyout clientkey.pem -out clientcert.pem -passout pass:changeit
2. openssl pkcs12 -export -inkey clientkey.pem -in clientcert.pem -out client.p12 -name myclientkey -passin pass:changeit -passout pass:changeit
3. keytool -importkeystore -destkeystore truststore.jks -deststorepass changeit -deststoretype jks -srckeystore client.p12 -srcstorepass changeit -srcstoretype pkcs12
4. keytool -list -keystore truststore.jks -storepass changeit -v
5. keytool -exportcert -alias myclientkey -storepass changeit -keystore truststore.jks -file client.cer
6. keytool -printcert -file client.cer
7. rm \*.pem \*.p12

Finally, import each key into the other’s keystore

1. keytool -import -noprompt -trustcacerts -alias myclientkey -file client.cer -keystore pmpKeystore.jks -storepass changeit
2. keytool -import -noprompt -trustcacerts -alias certificatekey -file service.cer -keystore truststore.jks -storepass changeit