**How To Create a Java Keystore via Keytool in FMW 11g/12c**All the commands below reference $MIDDLEWARE\_HOME for FMW 11g. If using FMW 12c, replace $MIDDLEWARE\_HOME with $ORACLE\_HOME.  
  
1. Create a directory, for example: $MIDDLEWARE\_HOME/keystores  
2. Run the following to set the environment on UNIX:

cd $MIDDLEWARE\_HOME/user\_projects/domains/<domain>/bin  
> . ./setDomainEnv.sh

Or on Windows:

cd %MIDDLEWARE\_HOME%\user\_projects\domains\<domain>\bin  
setDomainEnv.cmd

3. Create a keystore and private key, by executing the following command:

keytool -genkey -alias <alias> -keyalg RSA -keysize 1024 -sigalg SHA256withRSA -dname <dn> -keypass <password> -keystore <keystore> -storepass <password>

For example:

$MIDDLEWARE\_HOME/keystores> keytool -genkey -alias server\_cert -keyalg RSA -keysize 1024 -sigalg SHA256withRSA -dname "CN=server.uk.oracle.com,OU=Support,O=Oracle,L=Reading,ST=Berkshire,C=GB" -keypass welcome -keystore keystore.jks -storepass welcome

Example2 : keytool -genkey -alias q2c -keyalg RSA -keystore q2c.jks -keysize 2048 -keypass accenture -storepass Accenturewhere *server.uk.oracle.com* is the host.domain of the server.

Make sure you take note of the *-alias*, -*keypass*, and *-storepass* parameters as these will be required later in the process.  
  
4. At this point take a backup of the keystore e.g: keystore.jks  
5. To view the contents of the keystore created, execute the following command:

keytool -list -v -keystore <keystore> -storepass <password>

For example:

$MIDDLEWARE\_HOME/keystores> keytool -list -v -keystore keystore.jks -storepass welcome  
  
Keystore type: JKS

Keystore provider: SUN  
  
Your keystore contains 1 entry  
  
Alias name: server\_cert  
Creation date: Sep 13, 2010  
Entry type: PrivateKeyEntry  
Certificate chain length: 1  
Certificate[1]:  
Owner: CN=server.uk.oracle.com, OU=Support, O=Oracle, L=Reading, ST=Berkshire, C=GB  
Issuer: CN=server.uk.oracle.com, OU=Support, O=Oracle, L=Reading, ST=Berkshire, C=GB  
Serial number: 4c8e1ad5  
Valid from: Mon Sep 13 13:36:37 BST 2010 until: Sun Dec 12 12:36:37 GMT 2010  
Certificate fingerprints:  
MD5: 1A:4A:3B:42:7E:BD:94:65:67:0E:9B:02:28:90:D6:A8  
SHA1: C1:53:48:50:EB:F1:FD:A0:DC:28:9F:EF:3B:C8:FB:22:82:9F:8E:EE  
Signature algorithm name: SHA256withRSA  
Version: 3  
  
  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*  
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

If you are happy using a self signed certificate then this keystore is enough and you can move to step 12. otherwise if you need a proper certifictate signed by a real CA continuw below.

Example2: keytool -list -v -keystore keystore.jks -storepass accenture

6. Create a Certificate Signing Request (CSR) using the following command:

keytool -certreq -v -alias <alias> -file <filename> -sigalg SHA256withRSA -keypass <password> -storepass <password> -keystore <keystore>

For example:

MIDDLEWARE\_HOME/keystores/> keytool -certreq -v -alias server\_cert -file server.csr -sigalg SHA256withRSA -keypass welcome -storepass welcome -keystore keystore.jks

Make sure you use the same *-alias*, *-storepass* and *-keypass* passwords from Step 3.

Example2 : keytool -certreq -alias q2c -file request.pem -keystore q2c.jks  
  
The CSR (server.csr) created looks like this:

-----BEGIN NEW CERTIFICATE REQUEST-----  
MIIBtzCCASACAQAwdzELMAkGA1UEBhMCR0IxEjAQBgNVBAgTCUJlcmtzaGlyZTEQMA  
4GA1UEBxMHUmVhZGluZzEPMA0GA1UEChMGT3JhY2xlMRAwDgYDVQQLEwdTdXBwb3J0  
MR8wHQYDVQQDExZtYXJzaGFsbC51ay5vcmFjbGUuY29tMIGfMA0GCSqGSIb3DQEBA  
QUAA4GNADCBiQKBgQCEopgMZp1lI6jWXxb1rM1kWIc1l8bhiV/0UTcsdKzeaSHxbO  
SLO3Ed9kxNWAZgXaR9f5FBlwRJ+IR163e64v3SplHenxHfVRaHYWPZx4KlJz/6p  
Yd1fAlF0PdQm1DNoFtKmCHVk/cRuvGRpsp38l7K2mYlyQ+GxH38llS7g3owIDAQAB  
oAAwDQYJKoZIhvcNAQEFBQADgYEAD/sG1+rSI76OjihHg3WezT+VIbSRJxyly9nbx  
4uwXbDHh8DGgQLAXV51C9ioaMrm+dM0eygVDDMESXFxvJiYipS/pphgYt1xDBgnEH  
GcNiX3BnTaLNtzYlc5eAMsmbDlpk/qOxvQiH3bKN+UKYQlBXJZWPL6FusXu2LMTrk  
zsY=  
-----END NEW CERTIFICATE REQUEST-----

7. Send this CSR to a Certificate Authority (CA) of your choice.  
  
8. Once you have received the Certificate back you will need to import this along with the Trusted Root CA certificate(s) that signed it, into your keystore.

Note: If you are unsure what the correct Trusted Certificates are, the see [Note 1368940.1](https://support.oracle.com/epmos/faces/DocumentDisplay?parent=DOCUMENT&sourceId=1230333.1&id=1368940.1) How To Identify The Correct Trusted Root Certificate Authority Certificate(s) for a User Certificate?

Take the server certificate and save it a file called server.cer. Take the Certificate Authority's root certificate and save to a file called rootCA.cer in your keystore directory e.g $MIDDLEWARE\_HOME/keystores. Repeat this step for any more Root CA certificates in the chain e.g rootCA2.cer etc..  
  
9. Import the CA's root certificate into your keystore using the following command:

keytool -import -v -noprompt -trustcacerts -alias <alias>  
-file <rootca\_file> -keystore <keystore> -storepass <password>

For example:

$ORACLE\_MIDDLEWARE/keystores> keytool -import -v -noprompt -trustcacerts -alias rootcacert  
-file rootCA.cer -keystore keystore.jks -storepass welcome  
  
Certificate was added to keystore  
[Storing keystore.jks]

Repeat this for each Root CA certificate in the chain, and use a different alias each time.  
  
10. Import the Server Certificate into your keystore using the following command:

keytool -import -v -alias <alias> -file <server\_cert\_file> -keystore <keystore> -keypass <password> -storepass <password>

For example:

$MIDDLEWARE\_HOME/keytores> keytool -import -v -alias server\_cert -file server.cer -keystore keystore.jks -keypass welcome -storepass welcome  
Certificate reply was installed in keystore  
[Storing keystore.jks]

Make sure you use the same *-alias* from Step 3.  
  
11. To view the contents of the keystore, execute the following command:

keytool -list -v -keystore keystore.jks -storepass welcome

keytool -genkey -alias q2c -keyalg RSA -keystore q2c.jks -keysize 2048 -keypass accenture -storepass Accenture

keytool -certreq -alias q2c -file request.pem -keystore q2c.jks

export JAVA\_HOME=/accunix/java/jdk1.7.0\_40

keytool -list -v -keystore q2c.jks -keypass accenture -storepass Accenture

keytool -import -alias q2c -keystore q2c.jks -keypass accenture -storepass accenture -file ssl\_certificate.pem

keytool -list -v -keystore q2c.jks -keypass accenture -storepass accenture

[oracle@aiadv1ap01 soa\_domain]$ export JAVA\_HOME=/accunix/java/jdk1.7.0\_40

[oracle@aiadv1ap01 soa\_domain]$ keytool -list -v -keystore q2c.jks -keypass accenture -storepass accenture

Keystore type: JKS

Keystore provider: SUN

Your keystore contains 1 entry

Alias name: q2c

Creation date: Sep 23, 2016

Entry type: PrivateKeyEntry

Certificate chain length: 1

Certificate[1]:

Owner: CN=aiadv1ap01.rogers.com, OU=Q2C, O=Rogers Communications, L=Toronto, ST=Ontario, C=CA

Issuer: CN=aiadv1ap01.rogers.com, OU=Q2C, O=Rogers Communications, L=Toronto, ST=Ontario, C=CA

Serial number: 11830d08

Valid from: Fri Sep 23 10:57:38 EDT 2016 until: Thu Dec 22 09:57:38 EST 2016

Certificate fingerprints:

MD5: CD:F8:B8:BB:55:30:43:EE:D3:64:B5:1E:12:0B:3C:08

SHA1: 9B:83:00:F1:AC:40:70:1C:C5:75:48:92:BC:01:AB:53:F3:96:96:84

SHA256: 76:F0:35:EE:B3:02:78:D9:A3:99:77:13:E6:E4:81:D0:4B:E7:50:A3:0A:18:ED:BE:E7:98:C2:54:FB:47:4A:B8

Signature algorithm name: SHA256withRSA

Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false

SubjectKeyIdentifier [

KeyIdentifier [

0000: 2C A5 0F 62 0F 4F 21 B8 9B FF 3F 6B C8 48 51 DE ,..b.O!...?k.HQ.

0010: CC 90 F2 B2 ....

]

]

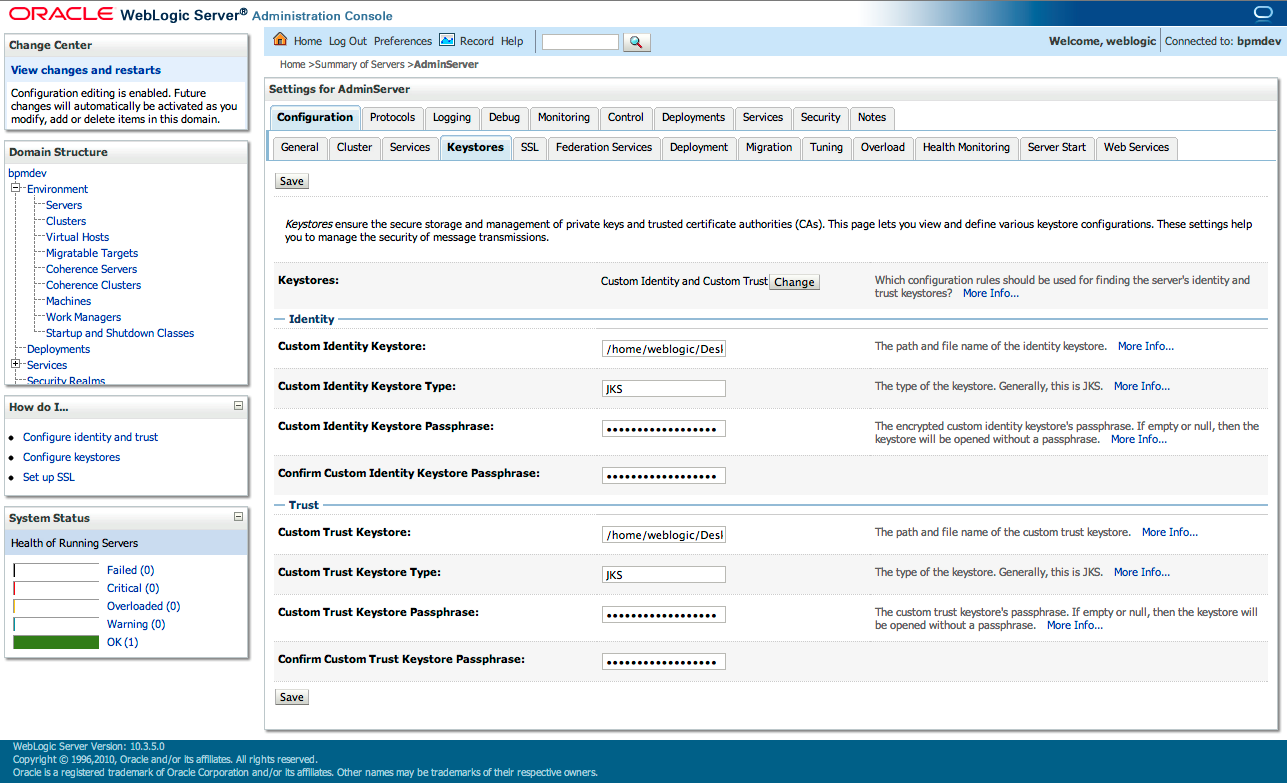
\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

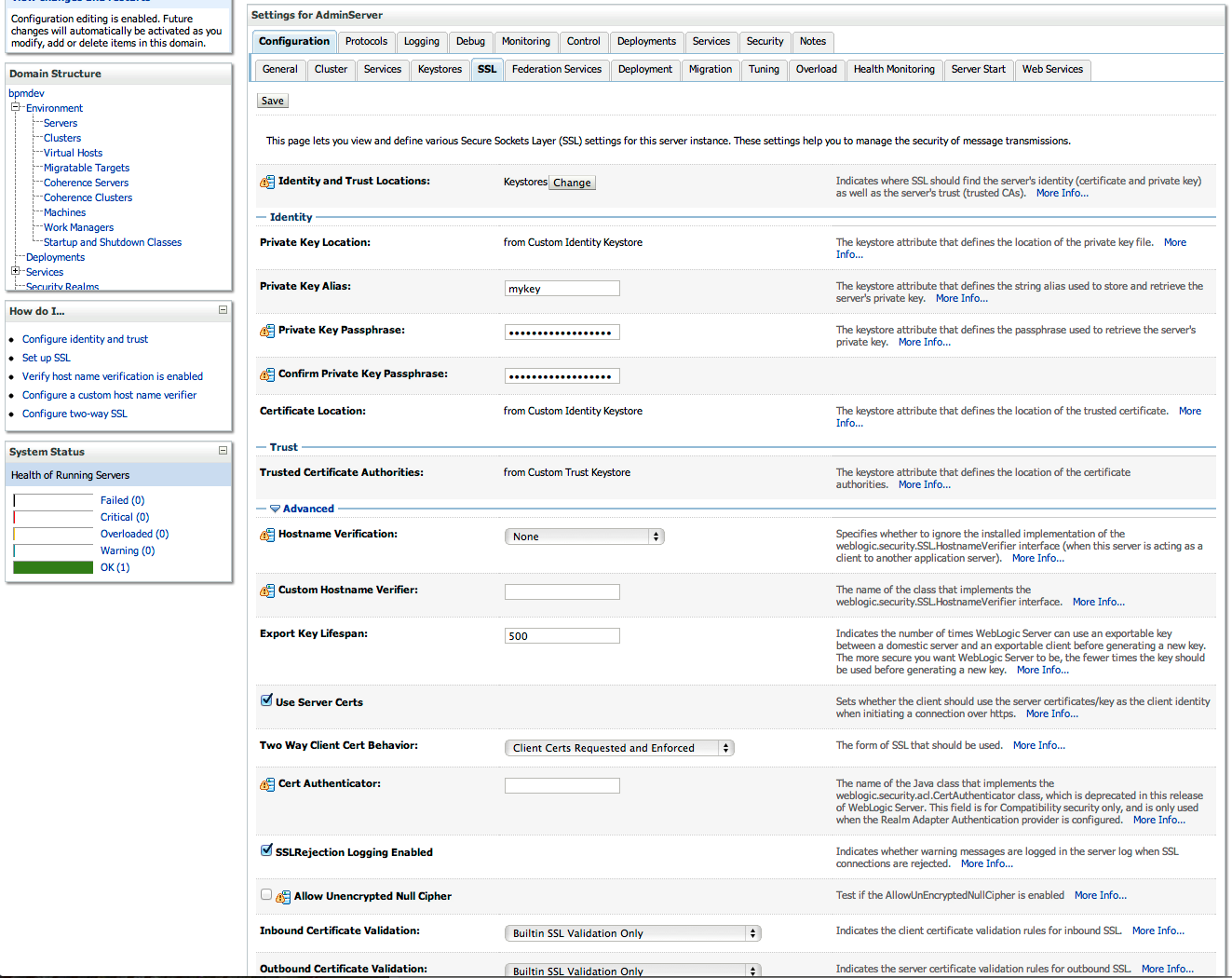
keytool -import -alias q2c -keystore q2c.jks -keypass accenture -storepass accenture -file Rapid\_SSL\_SHA256.pem -alias apublickey

#### Setup keystores and SSL on WebLogic

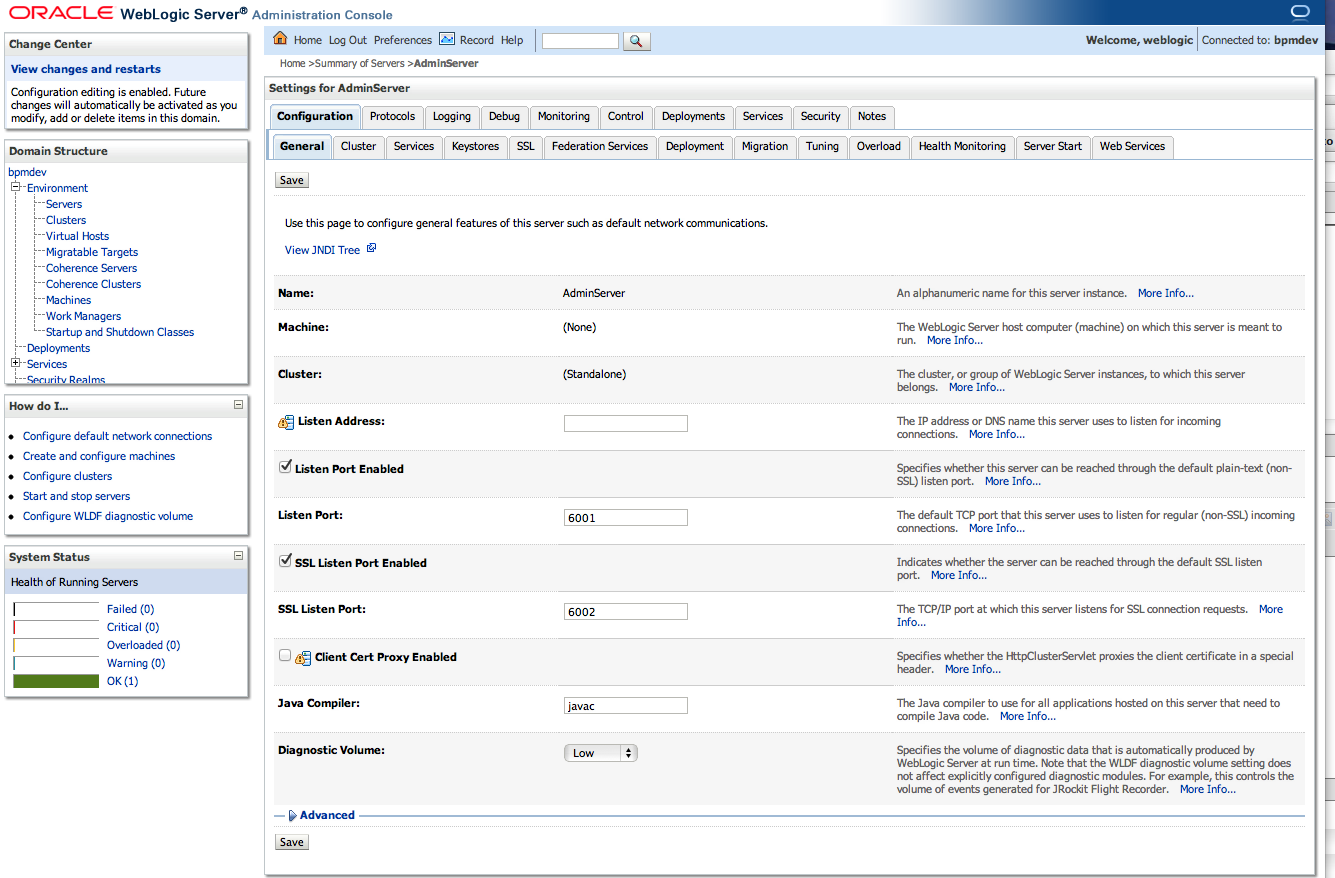
First, you will need to login to the WebLogic console, navigate to the server’s configuration->Keystore’s tab. Change the Keystores type to Custom Identity and Custom Trust and enter the rest of the fields.

Then you navigate to the SSL tab, enter the fields in the identity section and expand the Advanced section. Since I am using self signing cert on my VM enviornment, I disabled Hostname verification. On a production system, this should not be the case. I also enabled the option “Use Server Certs”, so that the application uses

the server cert to initiate https traffic (it is important to enable this in OSB).



Last, you enable SSL listening port in the Server’s configuration->General tab.



#### 3. Setup server to use 2 way SSL

If you follow the screen shot in previous step, you can see in the Server->Configuration->SSL->Advanced section, there is an option for Two Way Client Cert Behavior, you should set this to

Client Certs Requested and Enforced.

Repeat step 2 and 3 done on OSB. After all these configurations, you need to restart all the servers.

#### 4. Configure your SOA composite’s partner link to use 2 way SSL

You do this by modifying the composite.xml in your project, locate the partner’s link reference and add the property oracle.soa.two.way.ssl.enabled

|  |
| --- |
| <reference name=”callosb” ui:wsdlLocation=”helloword.wsdl”> < interface.wsdl interface=”http://www.examples.com/wsdl/HelloService.wsdl#wsdl.interface(Hello\_PortType)”/> < binding.ws port=”http://www.examples.com/wsdl/HelloService.wsdl#wsdl.endpoint(Hello\_Service/Hello\_Port)” location=”helloword.wsdl” soapVersion=”1.1″> < property name=”weblogic.wsee.wsat.transaction.flowOption” type=”xs:string” many=”false”>WSDLDriven</property> < property name=”oracle.soa.two.way.ssl.enabled”>true</property> < /binding.ws> < /reference> |

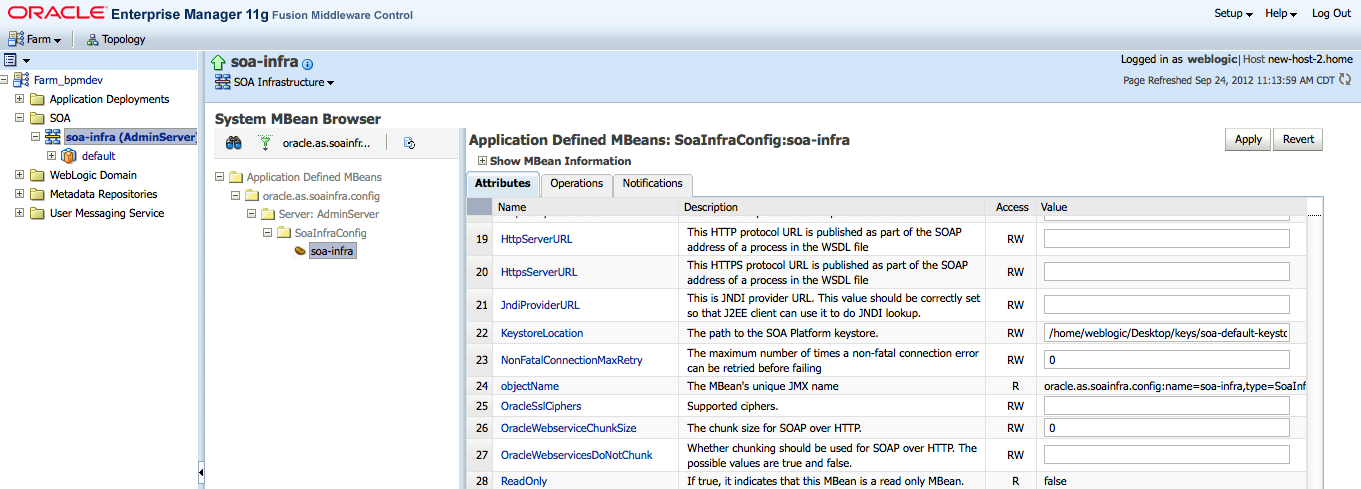
In OSB, you should have checked the HTTPS required flag in the proxy’s transport configuration. After this, rebuild the composite jar file and prepare to deploy in the EM console later.

#### 5. Configure SOA engine two ways SSL

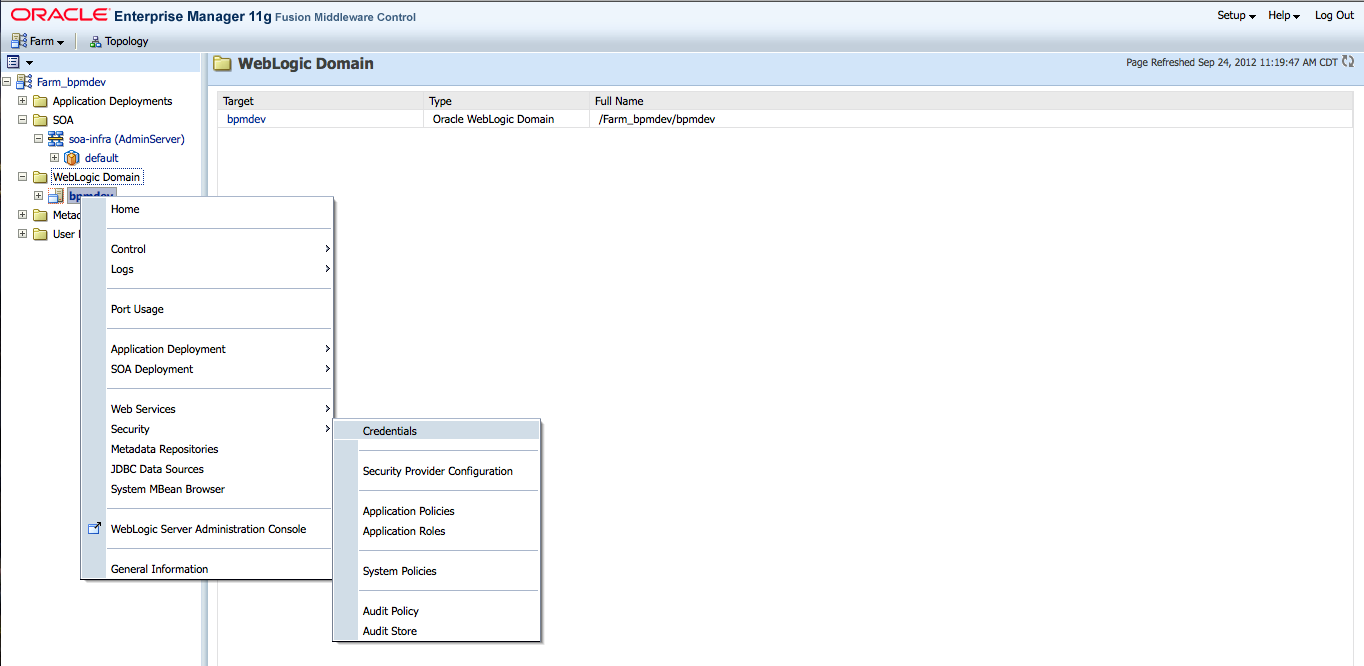
Oracle SOA Suite uses both Oracle WebLogic Server and Sun Secure Socket Layer (SSL) stacks for two-way SSL configurations.

* For the inbound web service bindings, Oracle SOA Suite uses the  
  Oracle WebLogic Server infrastructure and, therefore, the Oracle  
  WebLogic Server libraries for SSL. This is already done by step 2 and 3 in the previous section.
* For the outbound web service bindings, Oracle SOA Suite uses JRF HttpClient and, therefore, the Sun JDK libraries for SSL. You do this by configuring the SOA Engine in the Enterprise Manager Console, select soa-infra->SOA Administration->Common Properties

Then click at the link at the bottom of the page: “More SOA Infra Advances Infrastructure Configuration Properties” and then enter the full path of soa identity keystore in the value field of the KeyStoreLocation attribute.

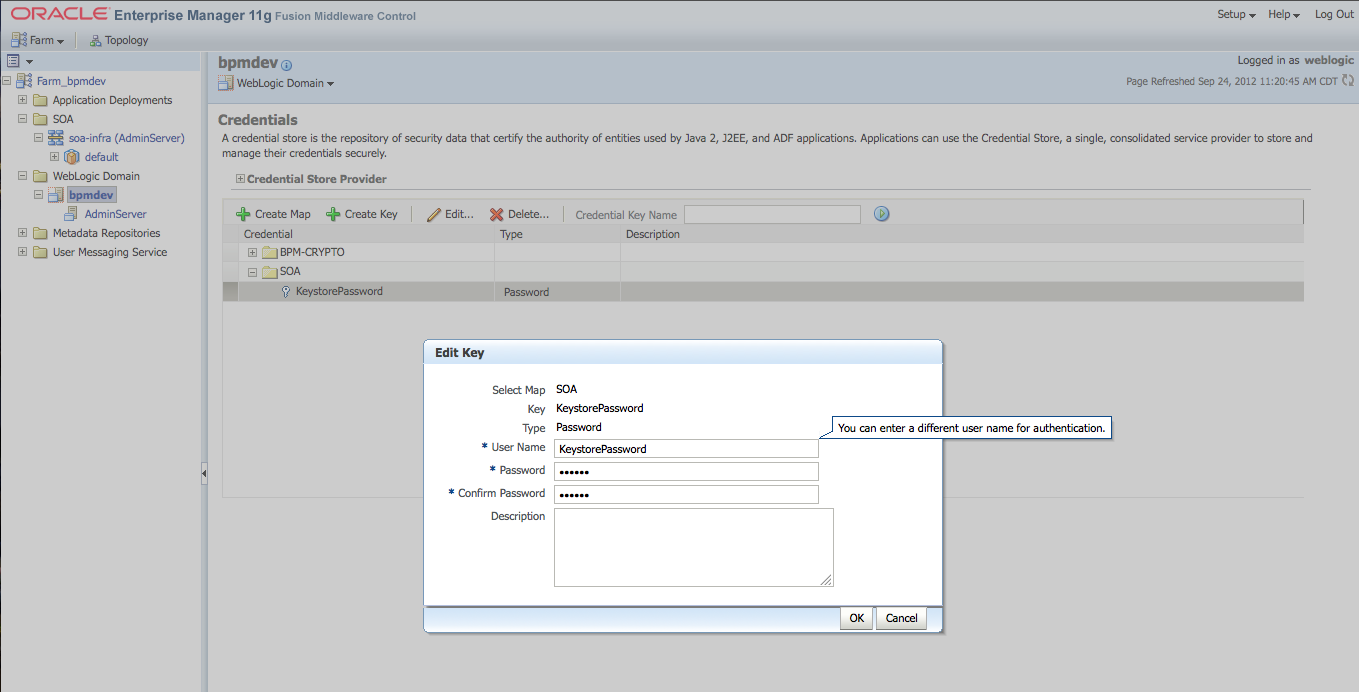


Click Apply and Return then navigate to the domain->security->credential.



Here, you provide the password to the keystore. Note: the alias of the certficate must be mykey as described in step 1, so you only need to provide the password to the identity keystore. You accomplish this by:

1. Click Create Map
2. In the Map Name field, enter SOA, and click OK
3. Click Create Key
4. Enter the following details where the password is the password for the SOA identity keystore.



#### Test and Trouble Shooting

Once the setup is complete and server restarted, you can deploy the composite usng the EM console and test it. In case of any errors, you should read the server log file to determine the cause of the error.

For example,

If you have not setup step 5 and test 2 way SSL, you will see this in the log when invoking OSB from BPEL:

|  |
| --- |
| java.lang.Exception: oracle.sysman.emSDK.webservices.wsdlapi.SoapTestException: oracle.fabric.common.FabricInvocationException: Unable to access the following endpoint(s): https://localhost.localdomain:7002/default/helloword ####<Sep 22, 2012 2:07:37 PM CDT> <Error> < oracle.soa.bpel.engine.ws> <rhel55> <AdminServer> < [ACTIVE] ExecuteThread: ‘1’ for queue: ‘weblogic.kernel.Default (self-tuning)’> <<anonymous>> <BEA1-0AFDAEF20610F8FD89C5> …………<11d1def534ea1be0:-4034173:139ef56d9f0:-8000-00000000000002ec> < 1348340857956> <BEA-000000> <got FabricInvocationExceptionsun.security.provider.certpath.SunCertPathBuilderException: unable to find valid certification path to requested target |

If you have not enable WebLogic SSL to use server certificate in the console and invoke SOA composite from OSB using two ways SSL, you will see this error:

|  |
| --- |
| ####<Sep 22, 2012 2:07:37 PM CDT> <Warning> < Security> <rhel55> <AdminServer> <[ACTIVE] ExecuteThread: ‘6’ for queue: ‘weblogic.kernel.Default (self-tuning)’> <<WLS Kernel>> <> <11d1def534ea1be0:-51f5c76a:139ef5e1e1a:-8000-00000000000000e2> < 1348340857776> <BEA-090485> <CERTIFICATE\_UNKNOWN alert was received from localhost.localdomain – 127.0.0.1. The peer has an unspecified issue with the certificate. SSL debug tracing should be enabled on the peer to determine what the issue is.> ####<Sep 22, 2012 2:07:37 PM CDT> <Warning> < Security> <rhel55> <AdminServer> <[ACTIVE] ExecuteThread: ‘6’ for queue: ‘weblogic.kernel.Default (self-tuning)’> <<WLS Kernel>> <> <11d1def534ea1be0:-51f5c76a:139ef5e1e1a:-8000-00000000000000e4> < 1348340857786> <BEA-090485> <CERTIFICATE\_UNKNOWN alert was received from localhost.localdomain – 127.0.0.1. The peer has an unspecified issue with the certificate. SSL debug tracing should be enabled on the peer to determine what the issue is.> ####<Sep 22, 2012 2:27:21 PM CDT> < Warning> <Security> <rhel55> <AdminServer> < [ACTIVE] ExecuteThread: ‘0’ for queue: ‘weblogic.kernel.Default (self-tuning)’> <<anonymous>> <> < 11d1def534ea1be0:-51f5c76a:139ef5e1e1a:-8000-0000000000000124> < 1348342041926> <BEA-090497> <HANDSHAKE\_FAILURE alert received from localhost – 127.0.0.1. Check both sides of the SSL configuration for mismatches in supported ciphers, supported protocol versions, trusted CAs, and hostname verification settings.> |

References

1. <http://docs.oracle.com/cd/E23943_01/admin.1111/e10226/soacompapp_secure.htm#CHDCFABB> Section 5.6.4
2. http://docs.oracle.com/cd/E23943\_01/web.1111/e13707/ssl.htm#i1200848