Self-signed certificate and configure Custom Identity and Custom Trust with

Intro about the certificates:

Many organizations are tempted to use self-signed SSL Certificates instead of those issued and verified by a trusted Certificate Authority mainly because of the price difference. Unlike CA issued certificates, self-signed certificates are free of charge. What most users are not aware of is that self-signed certificates can end up costing them more in the long run.

While self-signed SSL Certificates also encrypt customers' log in and other personal account credentials, they prompt most web servers to display a security alert because the certificate was not verified by a trusted Certificate Authority. Often the alerts advise the visitor to abort browsing the page for security reasons.

Intro to SHA:

SHA - standing for secure hash algorithm –

SHA, which stands for secure hash algorithm, is a cryptographic hashing algorithm used to determine the integrity of a particular piece of data. Variations of this algorithm are often used by SSL certificate authorities to sign certificates. **This algorithm help ensures that your website's data is not modified or tampered with**. It does so by generating unique hash values from any particular file / variation of a file. Based on these hash values, it can be determined whether or not the file has been altered by comparing the expected hash value to the hash value received.

As computers become more powerful, the SHA hash sizes are increasing to help better improve security and make it harder for attackers to decrypt hashes. The secure hash algorithm originally started out as SHA0 (a 160-bit hash published in 1993). As of when this article was published, there is currently a much more powerful SHA known as SHA3 (a 1600-bit hash).

There are, therefore, several versions of SHA: SHA0 (obsolete because vulnerable), SHA1 (the most popular one), SHA2 (the one we are interested in) and finally SHA3 introduced in 2012.

Intro SHA2 :

SHA2, not often used for now, is the successor of SHA1 and gathered 4 kinds of hash functions: SHA224, SHA256, SHA384 and SHA512.

It works the same way than SHA1 but is stronger and generates a longer hash.

SHA1 vs SHA256 :

will focus mainly on the differences that exist between SHA1 vs SHA256. SHA2 is the successor of SHA1 and is commonly used by many SSL certificate authorities. There are currently 6 different SHA2 variants including:

Self-signed certificate and configure Custom Identity and Custom Trust with

- SHA-224
- SHA-256
- SHA-384
- SHA-512
- SHA-512/224
- SHA-512/256

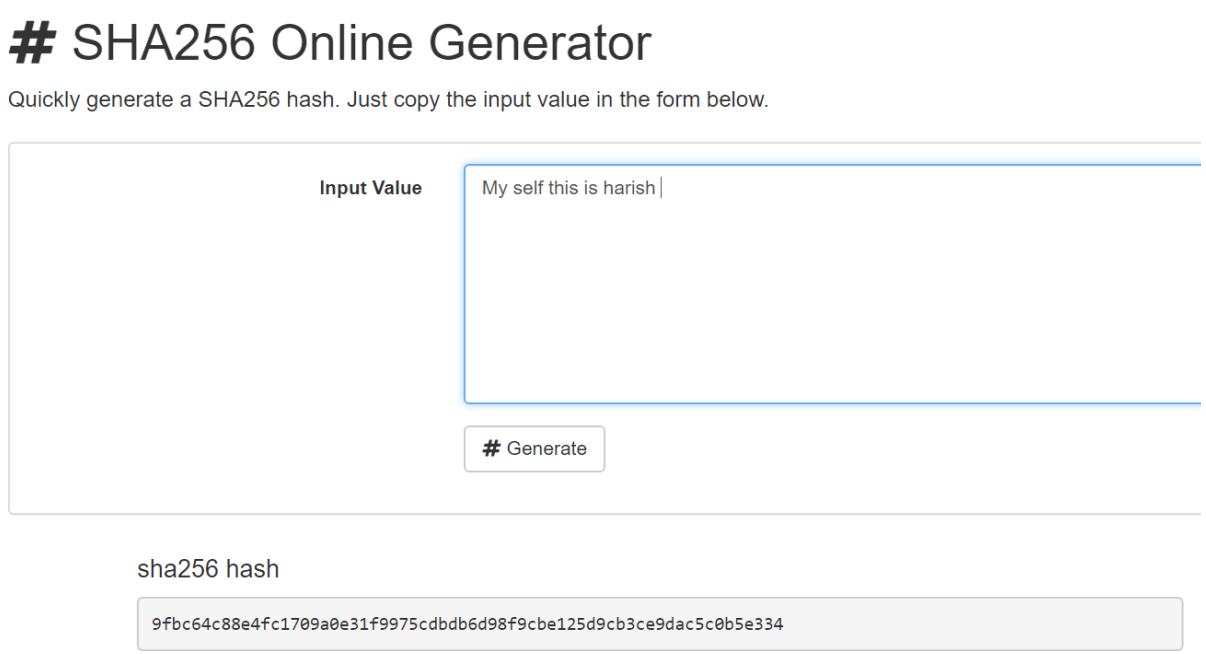
These variations differ in terms of output size, internal state size, block size, message size, and rounds. To compare the differences that exist between the SHA1 vs SHA256 algorithms, consider the following SHA comparison information from Wikipedia.

Due to SHA1's smaller bit size, it has become **more susceptible to attacks** which therefore led to its deprecation from SSL certificate issuers in January 2016. An example of the difference in size between SHA1 vs SHA256 can be seen in the following example hashes:

- **SHA1** – da39a3ee5e6b4b0d3255bfef95601890afd80709
- **SHA256** – e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855

With our online hash generator tool, you can quickly generate an SHA256 hash for any string or input value. Simply enter a string value into the input box and select Generate. The tool will then **generate a unique 64-digit hash** for the value you specified.

Below is the example:



Self-signed certificate and configure Custom Identity and Custom Trust with

Will continue our self-certificates with sha2 Algorithm:

To generate a identity Keystore with the validity of the certificate:

Step 1:

To create a 2048 bit SHA2/SHA256 certificate use the following command :

./keytool -genkey -alias hintkey -keyalg RSA -keysize 2048 -sigalg SHA256withRSA -validity 365 -keypass password -keystore identity.jks -storepass password

```
[root@hint102 bin]# ./keytool -genkey -alias hintkey -keyalg RSA -keysize 2048 -sigalg SHA256withRSA -validity 365 -keypass password -ke
ystore identity.jks -storepass password
What is your first and last name?
  [Unknown]:  HINT
What is the name of your organizational unit?
  [Unknown]:  IT TRAININGS
What is the name of your organization?
  [Unknown]:  HI N Technologies
What is the name of your City or Locality?
  [Unknown]:  Bangalore
What is the name of your State or Province?
  [Unknown]:  KA
What is the two-letter country code for this unit?
  [Unknown]:  IN
Is CN=HINT, OU=IT TRAININGS, O=HI N Technologies, L=Bangalore, ST=KA, C=IN correct?
  [no]:  yes

[root@hint102 bin]#
```

Step 2:

Now we are exporting Identity file to the cert file

./keytool -export -alias hintkey -file hint.cer -keystore identity.jks -storepass password

```
[root@hint102 bin]# ./keytool -export -alias hintkey -file hint.cer -keystore identity.jks -storepass password
Certificate stored in file <hint.cer>
[root@hint102 bin]#
```

Step 3:

Now Import csr file to the trust Keystore:

./keytool -import -alias hintkey -file hint.cer -keystore trust.jks -storepass password

```
[root@hint102 bin]# ./keytool -export -alias hintkey -file hint.cer -keystore identity.jks -storepass password
Certificate stored in file <hint.cer>
[root@hint102 bin]# ./keytool -import -alias hintkey -file hint.cer -keystore trust.jks -storepass password
Owner: CN=HINT, OU=IT TRAININGS, O=HI N Technologies, L=Bangalore, ST=KA, C=IN
Issuer: CN=HINT, OU=IT TRAININGS, O=HI N Technologies, L=Bangalore, ST=KA, C=IN
Serial number: 5997a3be
Valid from: Sat Aug 19 08:04:38 IST 2017 until: Sun Aug 19 08:04:38 IST 2018
Certificate fingerprints:
         MD5:  24:BB:4E:4E:34:89:4B:E2:53:D5:35:79:EC:8B:29:88
         SHA1: 30:D0:69:D0:98:7C:0F:8E:00:91:CF:F0:FB:5B:42:A5:40:50:44:63
         Signature algorithm name: SHA256withRSA
         Version: 3
Trust this certificate? [no]:  yes
Certificate was added to keystore
[root@hint102 bin]#
```

Self-signed certificate and configure Custom Identity and Custom Trust with

To see the contents of the keystore use the following command :

keytool -list -v -keystore identity.jks -storepass password

```
[root@hint102 bin]# ./keytool -list -v -keystore identity.jks -storepass password

Keystore type: JKS
Keystore provider: SUN

Your keystore contains 1 entry

Alias name: hintkey
Creation date: Aug 19, 2017
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=HINT, OU=IT TRAININGS, O=HI N Technologies, L=Bangalore, ST=KA, C=IN
Issuer: CN=HINT, OU=IT TRAININGS, O=HI N Technologies, L=Bangalore, ST=KA, C=IN
Serial number: 5997a3be
Valid from: Sat Aug 19 08:04:38 IST 2017 until: Sun Aug 19 08:04:38 IST 2018
Certificate fingerprints:
         MD5:  24:BB:4E:4E:34:89:4B:E2:53:D5:35:79:EC:8B:29:88
         SHA1: 30:D0:69:D0:98:7C:0F:8E:00:91:CF:F0:FB:5B:42:A5:40:50:44:63
         Signature algorithm name: SHA256withRSA
         Version: 3


*******************************************
*******************************************
```

To see the contents of an individual certificate (like root.cer in our case):

keytool -printcert -file root.cer

```
[root@hint102 bin]# ./keytool -printcert -file root.cer
keytool error: java.io.FileNotFoundException: root.cer (No such file or directory)
[root@hint102 bin]# ./keytool -printcert -file hint.cer
Owner: CN=HINT, OU=IT TRAININGS, O=HI N Technologies, L=Bangalore, ST=KA, C=IN
Issuer: CN=HINT, OU=IT TRAININGS, O=HI N Technologies, L=Bangalore, ST=KA, C=IN
Serial number: 5997a3be
Valid from: Sat Aug 19 08:04:38 IST 2017 until: Sun Aug 19 08:04:38 IST 2018
Certificate fingerprints:
         MD5:  24:BB:4E:4E:34:89:4B:E2:53:D5:35:79:EC:8B:29:88
         SHA1: 30:D0:69:D0:98:7C:0F:8E:00:91:CF:F0:FB:5B:42:A5:40:50:44:63
         Signature algorithm name: SHA256withRSA
         Version: 3
[root@hint102 bin]#
```

Self-signed certificate and configure Custom Identity and Custom Trust with

Step4:

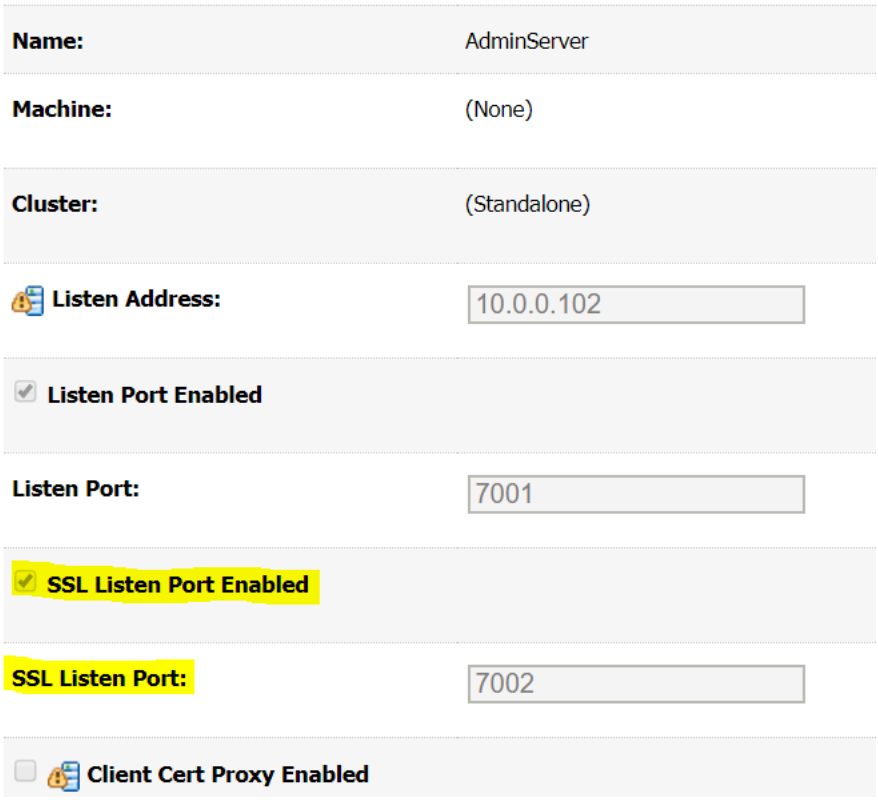
Copy the keystore files in the Cert location :

```
[root@hint102 bin]# cd -
/Oracle/Hint_certificates
[root@hint102 Hint_certificates]# ls -lrt
total 12
-rw-r--r--. 1 root root 2217 Aug 19 08:04 identity.jks
-rw-r--r--. 1 root root  864 Aug 19 08:07 hint.cer
-rw-r--r--. 1 root root  928 Aug 19 08:09 trust.jks
[root@hint102 Hint_certificates]#
```

Configure Custom Identity and Custom Trust with WebLogic Server :

Step 5 :

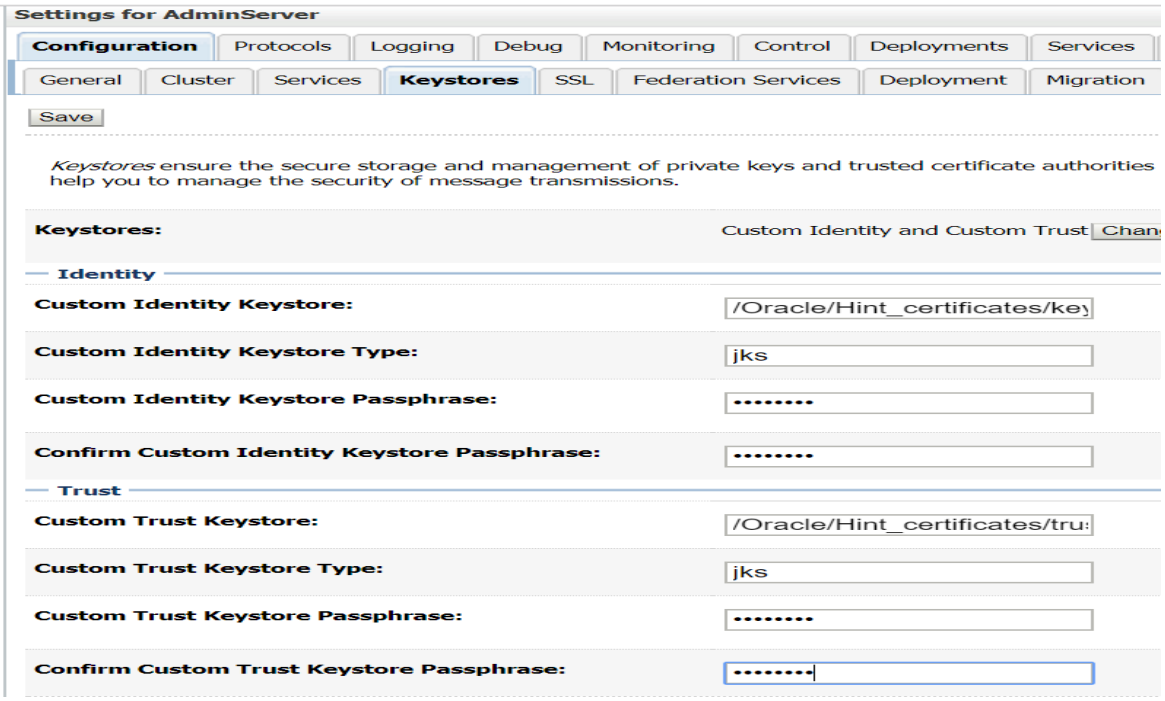
Login to Weblogic Admin console --> Environment --> Servers --> < server_name_where_ssl_has_to_be_configured > --> Configuration -> General --> SSL Listen Port Enabled (Check)



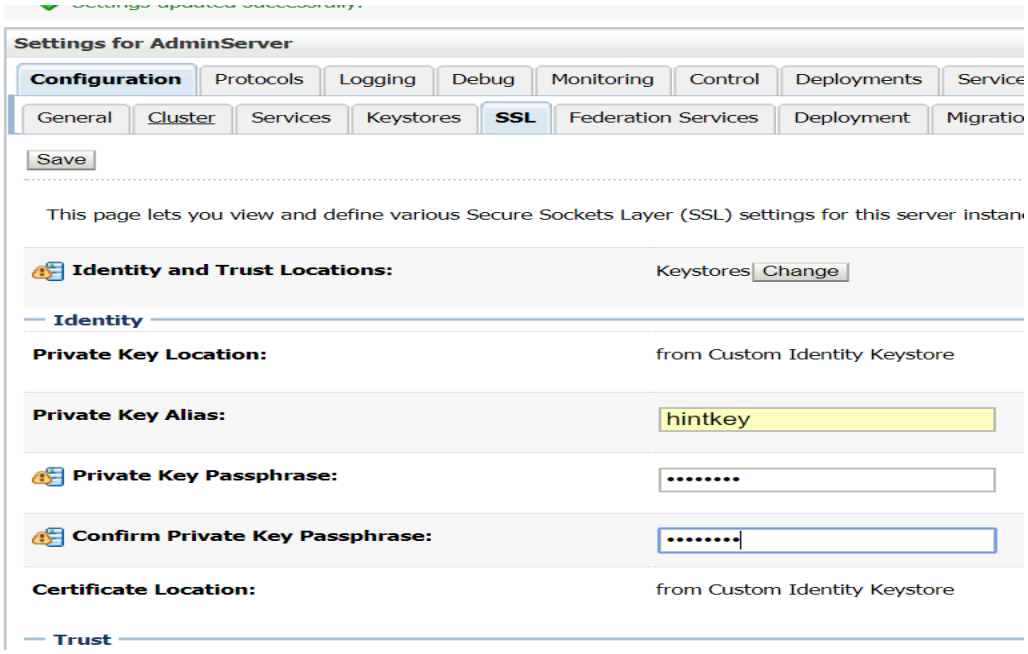
Self-signed certificate and configure Custom Identity and Custom Trust with

Step 6:

Go to the keystore and Add the certificates :



Now go the ssl and add the and add :



Step 7 :

Restart the JVM in the running log you can find that ssl port is listening .

```
tificates/trust.jks.>
<Aug 19, 2017 8:32:40 AM IST> <Notice> <Server> <BEA-002613> <Channel "sip" is now listening on 10.0.0.102:5060 for protocols sip.>
<Aug 19, 2017 8:32:40 AM IST> <Notice> <Server> <BEA-002613> <Channel "sips" is now listening on 10.0.0.102:5061 for protocols sips.>
<Aug 19, 2017 8:32:40 AM IST> <Notice> <Server> <BEA-002613> <Channel "Default" is now listening on 10.0.0.102:2700 for protocols iiop,
t3, ldap, snmp, http.>
<Aug 19, 2017 8:32:40 AM IST> <Notice> <Server> <BEA-002613> <Channel "DefaultSecure" is now listening on 10.0.0.102:7002 for protocols
iiops, t3s, ldaps, https.>
<Aug 19, 2017 8:32:40 AM IST> <Notice> <WebLogicServer> <BEA-000329> <Started WebLogic Admin Server "AdminServer" for domain "HINT027_do
main" running in Production Mode>
<Aug 19, 2017 8:32:40 AM IST> <Notice> <WLSS Transport> <BEA-330687> <Thread "STR Message processor (Transport UDP)" is listening on por
```

Step 8:

Now access the console with the https port. And check the ssl validity.