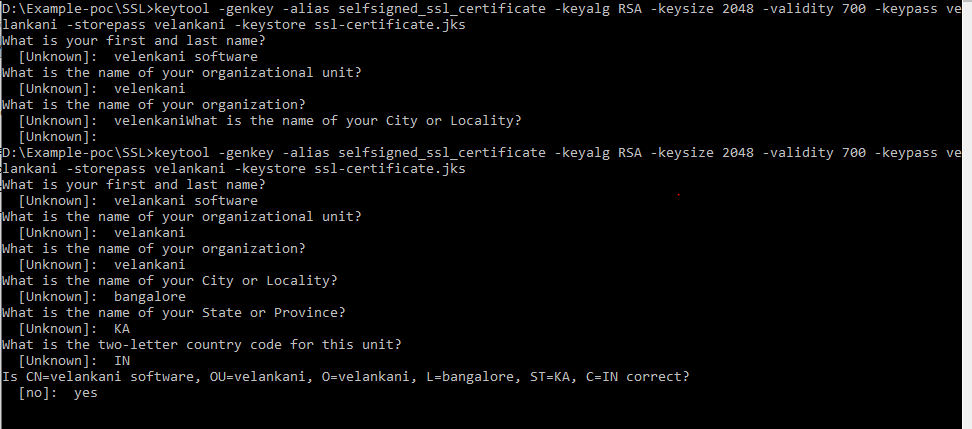
**HTTPS CONFIGURATION**

**1. Generating Certificate**

**1. Command to Generate Certificate**

**A. keytool -genkey -alias selfsigned\_ssl\_certificate -keyalg RSA -keysize 2048 -validity 700 -keypass changeit -storepass changeit -keystore ssl-certificate.jks**

* -genkey – is the keytool command to generate the certificate.
* -alias **selfsigned\_ssl\_certificate** – indicates the alias of the certificate, which is used by SSL/TLS layer
* -keyalg RSA -keysize 2048 -validity 700 – are self descriptive parameters indicating the crypto algorithm, keysize and certificate validity.
* -keypass **changeit** -storepass **changeit** – are the passwords of our truststore and keystore
* -keystore **ssl-server.jks** – is the actual keystore where the certificate and public/private key will be stored.

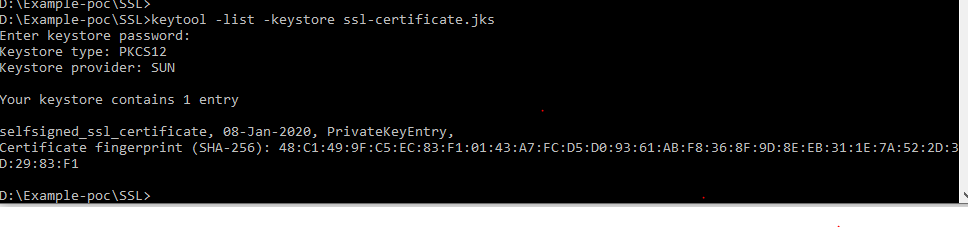
****

**B. keytool -list -keystore ssl-certificate.jks**

**ssl-certificate.jks----> we can see**

**Keystore type: PKCS12**

**Keystore provider: SUN**

****

**C**. **keytool -list -rfc -keystore ssl-certificate.jks**

**We can see ssl certificate in command prompt.**

**2. MAS Configuration Changes**

* **Generated jks file we need to keep in config location.**
* **rest-config.properties we need to add this four properties**
  + server.ssl.key.password=changeit
  + server.ssl.key.store.password=changeit
  + server.ssl.key.store.file=config/ssl-certificate.jks
  + server.ssl.key-store-type=PKCS12
* **RestConstants java class add this four variables.**
* **public** **static** String *SERVER\_SSL\_KEY\_PASSWORD* = "server.ssl.key.password";
* **public** **static** Strin *SERVER\_SSL\_KEY\_STORE\_PASSWORD*="server.ssl.key.store.password";
* **public** **static** String *SERVER\_SSL\_KEY\_STORE\_FILE*= "server.ssl.key.store.file";
* **public** **static** String *SERVER\_SSL\_KEY\_STORE\_TYPE* = "server.ssl.key-store- type";
* **nocvue-mas-service-> model StartRestService java class Https code we added while starting server.**

**nocvue-mas-service--> Model**

StartRestService.java class

Remove below lines in this StartRestService.java file

RestPropertyReader propertyReader = RestPropertyReader.getInstance();

int masPort = propertyReader.getInt(RestConstants.MAS\_PORT);

// Setup Threadpool

QueuedThreadPool threadPool = new QueuedThreadPool();

threadPool.setMinThreads(propertyReader.getInt(RestConstants.JETTY\_MIN\_THREADS));

threadPool.setMaxThreads(propertyReader.getInt(RestConstants.JETTY\_MAX\_THREADS));

threadPool.setIdleTimeout(propertyReader.getInt(RestConstants.JETTY\_IDLE\_TIMEOUT));

// Server server = new Server(masPort);

Server server = new Server(threadPool);

RestConstants.CONNECTION\_TIMEOUT = propertyReader.getLong(RestConstants.CONNECTION\_TIMEOUT\_KEY);

RestConstants.CONNECTION\_TIMEOUT\_DIFF = propertyReader.getLong(RestConstants.CONNECTION\_TIMEOUT\_DIFF\_KEY);

RestConstants.RECEIVE\_TIMEOUT = propertyReader.getLong(RestConstants.RECEIVE\_TIMEOUT\_KEY);

ServerConnector http = new ServerConnector(server);

http.setPort(masPort);

http.setIdleTimeout(RestConstants.CONNECTION\_TIMEOUT-RestConstants.CONNECTION\_TIMEOUT\_DIFF);

server.addConnector(http);

**Add this method call removed line place.**

Server server = getHttpsServer();

**Add this method at ending start(){}**

**public** Server getHttpsServer() **throws** FileNotFoundException, IOException {

RestPropertyReader propertyReader = RestPropertyReader.*getInstance*();

**int** masPort = propertyReader.getInt(RestConstants.*MAS\_PORT*);

// Setup Threadpool

QueuedThreadPool threadPool = **new** QueuedThreadPool();

threadPool.setMinThreads(propertyReader.getInt(RestConstants.*JETTY\_MIN\_THREADS*));

threadPool.setMaxThreads(propertyReader.getInt(RestConstants.*JETTY\_MAX\_THREADS*));

threadPool.setIdleTimeout(propertyReader.getInt(RestConstants.*JETTY\_IDLE\_TIMEOUT*));

Server server = **new** Server(threadPool);

RestConstants.*CONNECTION\_TIMEOUT* = propertyReader.getLong(RestConstants.***CONNECTION\_TIMEOUT\_KEY***);

RestConstants.*CONNECTION\_TIMEOUT\_DIFF* = propertyReader.getLong(RestConstants.***CONNECTION\_TIMEOUT\_DIFF\_KEY***);

RestConstants.*RECEIVE\_TIMEOUT* = propertyReader.getLong(RestConstants.***RECEIVE\_TIMEOUT\_KEY***);

String SERVER\_SSL\_KEY\_PASSWORD = propertyReader.getString(RestConstants.*SERVER\_SSL\_KEY\_PASSWORD*);

String SERVER\_SSL\_KEY\_STORE\_PASSWORD = propertyReader.getString(RestConstants.*SERVER\_SSL\_KEY\_STORE\_PASSWORD*);

String SERVER\_SSL\_KEY\_STORE\_FILE = propertyReader.getString(RestConstants.*SERVER\_SSL\_KEY\_STORE\_FILE*);

Path keystorePath = Paths.*get*(SERVER\_SSL\_KEY\_STORE\_FILE).toAbsolutePath();

**if** (!Files.*exists*(keystorePath))

**throw** **new** FileNotFoundException(keystorePath.toString());

HttpConfiguration httpsConfig = **new** HttpConfiguration();

httpsConfig.setSecureScheme("https");

httpsConfig.setSecurePort(masPort);

httpsConfig.addCustomizer(**new** SecureRequestCustomizer());

// SSL Context Factory for HTTPS and HTTP/2

SslContextFactory sslContextFactory = **new** SslContextFactory();

sslContextFactory.setKeyStorePath(keystorePath.toString());

sslContextFactory.setKeyStorePassword(SERVER\_SSL\_KEY\_PASSWORD);

sslContextFactory.setKeyManagerPassword(SERVER\_SSL\_KEY\_STORE\_PASSWORD);

// SSL Connection Factory

SslConnectionFactory ssl = **new** SslConnectionFactory(sslContextFactory, HttpVersion.***HTTP\_1\_1***.asString());

// HTTP/2 Connector

ServerConnector http2Connector = **new** ServerConnector(server, ssl, **new** HttpConnectionFactory(httpsConfig));

http2Connector.setIdleTimeout(RestConstants.*CONNECTION\_TIMEOUT*-RestConstants.*CONNECTION\_TIMEOUT\_DIFF*);

http2Connector.setPort(masPort);

server.addConnector(http2Connector);

**return** server;

}