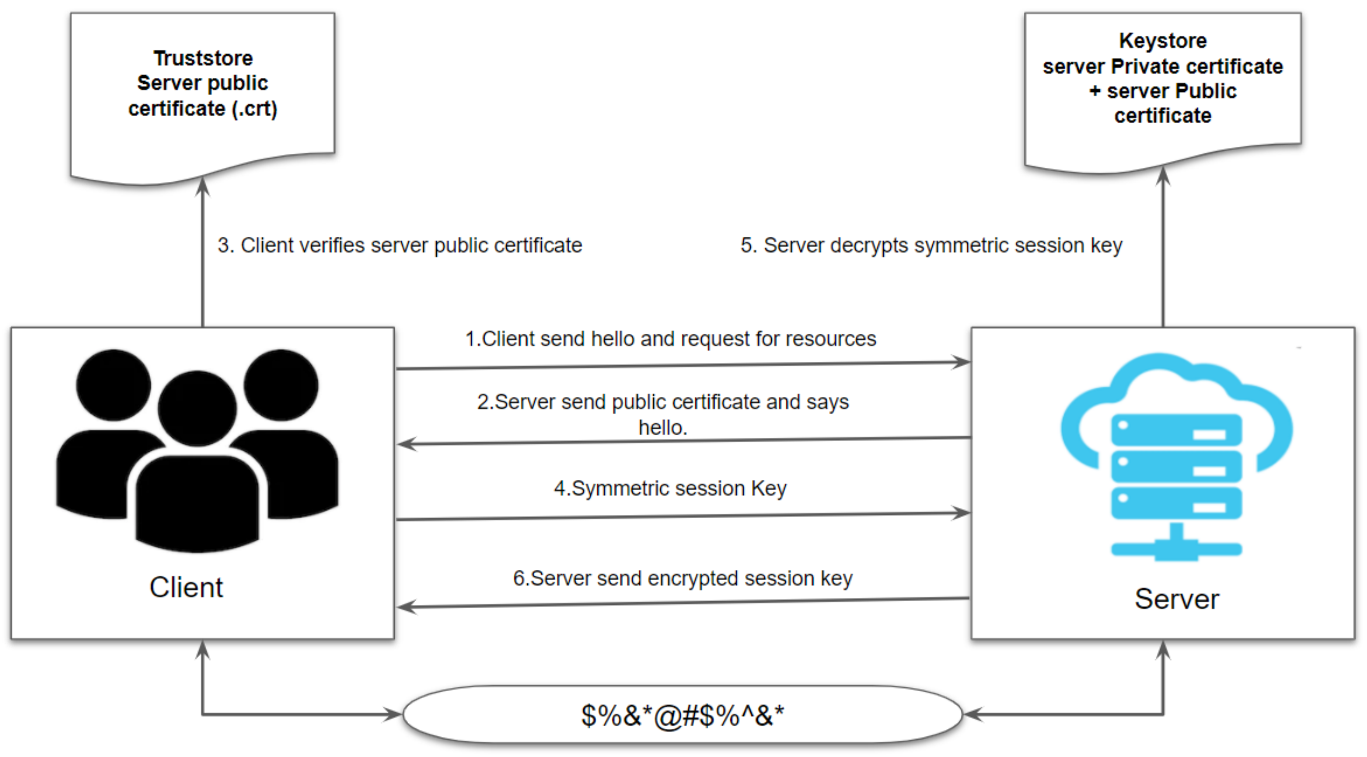
Introduction

In this article, we will be going to learn how to set up the one-way SSL and two-way SSL for MuleSoft applications. In one way SSL, the client always verifies the server certificates and the server never verifies the client certificates whereas in two-way SSL client verifies the server certificates and server verifies the client certificates. Sometimes two-way SSL is also known as Mutual Authentication.

One Way SSL

As mentioned above in one way SSL only client verifies the server certificates. At the server end, there will be a Keystore that will hold the private and public certificate of the server whereas, at the client end, there will be a truststore that will hold the public certificate of the server.

- Clients will send Hello and request for the resources on the secure HTTPS protocol.
- The server will respond with its public certificate (.crt) and send Hello.
- The client will verify the server public certificate in its truststore.
- The client sends back symmetric session key generated using the server public certificate.
- The server will decrypt the symmetric session key using its private certificate and send back the encrypted session key to the client for establishing a secure connection.


Truststore
Server public
certificate (.crt)
Keystore
server Private certificate
+ server Public
certificate
3. Client verifies server public certificate
5. Server decrypts symmetric session key
1.Client send hello and request for resources
2.Server send public certificate and says hello.
4.Symmetric session Key
6.Server send encrypted session key
Client
Server
$%&*@#$%^&*

Implementing One Way SSL For MuleSoft Application

For generating Keystore and truststore, we can use OpenSSL or keytool utility. For this article, we will be using keytool which is part of your JDK.

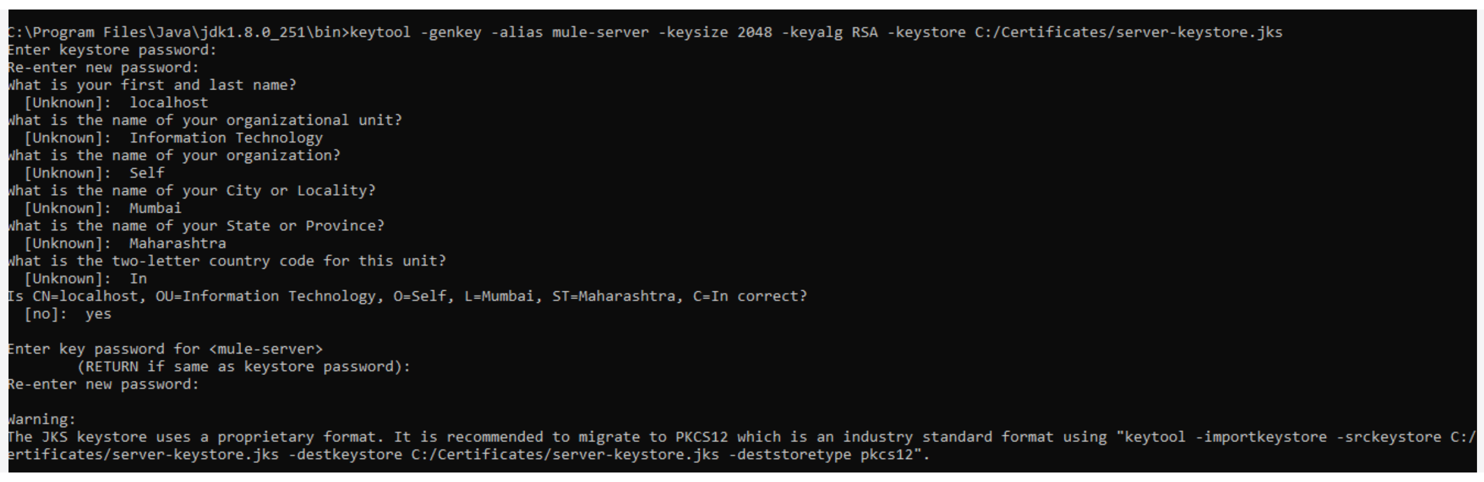
Step 1: Generate Server Keystore

For generating Keystore, we will be using the below command.

Plain Text
```
keytool -genkey -alias mule-server -keysize 2048 -keyalg RSA -keystore C:/Certificates/server-keystore.jks
```

Once we execute the above command, it will ask for the Keystore password, first and last name including other details and the key password.

alias in the above command is used to search the certificate in the Keystore as there can be multiple certificates in the same Keystore and alias is useful in identifying the right certificate.



```
C:\Program Files\Java\jdk1.8.0_251\bin>keytool -genkey -alias mule-server -keysize 2048 -keyalg RSA -keystore C:/Certificates/server-keystore.jks
Enter keystore password:
Re-enter new password:
What is your first and last name?
  [Unknown]:  localhost
What is the name of your organizational unit?
  [Unknown]:  Information Technology
What is the name of your organization?
  [Unknown]:  Self
What is the name of your City or Locality?
  [Unknown]:  Mumbai
What is the name of your State or Province?
  [Unknown]:  Maharashtra
What is the two-letter country code for this unit?
  [Unknown]:  In
Is CN=localhost, OU=Information Technology, O=Self, L=Mumbai, ST=Maharashtra, C=In correct?
  [no]:  yes

Enter key password for <mule-server>
        (RETURN if same as keystore password):
Re-enter new password:

Warning:
The JKS keystore uses a proprietary format. It is recommended to migrate to PKCS12 which is an industry standard format using "keytool -importkeystore -srckeystore C:/
ertificates/server-keystore.jks -destkeystore C:/Certificates/server-keystore.jks -deststoretype pkcs12".
```

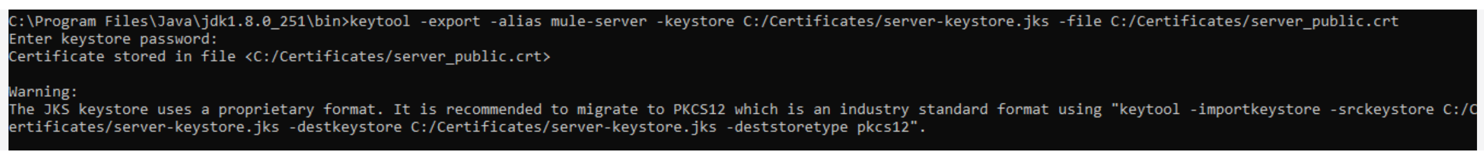
Step 2: Export the Public Certificate From Server Keystore

We will be requiring the server public certificate to be installed in the client truststore. So we need to extract the public key from the server Keystore.

For extracting the server public certificate, we will be using the below export command.

Plain Text
```
keytool -export -alias mule-server -keystore C:/Certificates/server-keystore.jks -file C:/Certificates/server_public.crt
```

This command will export the server public certificate and we will import that server public certificate in the client truststore.



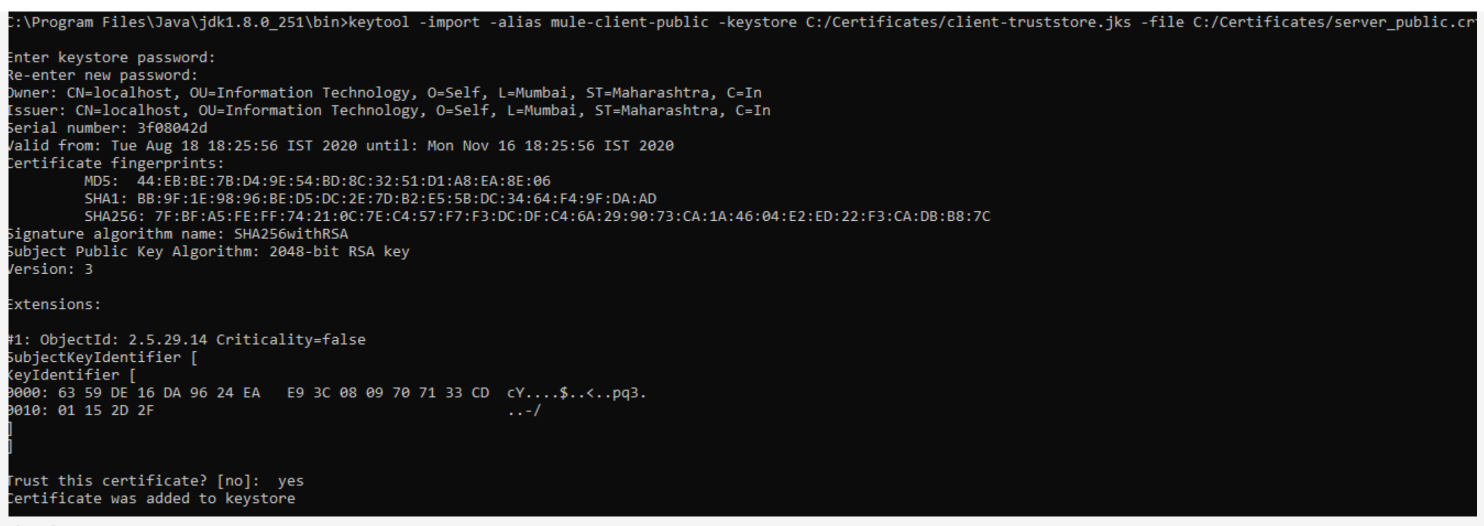
```
C:\Program Files\Java\jdk1.8.0_251\bin>keytool -export -alias mule-server -keystore C:/Certificates/server-keystore.jks -file C:/Certificates/server_public.crt
Enter keystore password:
Certificate stored in file <C:/Certificates/server_public.crt>

Warning:
The JKS keystore uses a proprietary format. It is recommended to migrate to PKCS12 which is an industry standard format using "keytool -importkeystore -srckeystore C:/C
ertificates/server-keystore.jks -destkeystore C:/Certificates/server-keystore.jks -deststoretype pkcs12".
```

Step 3: Import Server Public Certificate Into Client Truststore

For importing the server public key into the client truststore, we will be using the below command.

Plain Text
```
keytool -import -alias mule-client-public -keystore C:/Certificates/client-truststore.jks -file C:/Certificates/server_public.crt
```

This above command will generate the client's truststore and import the server public certificate.



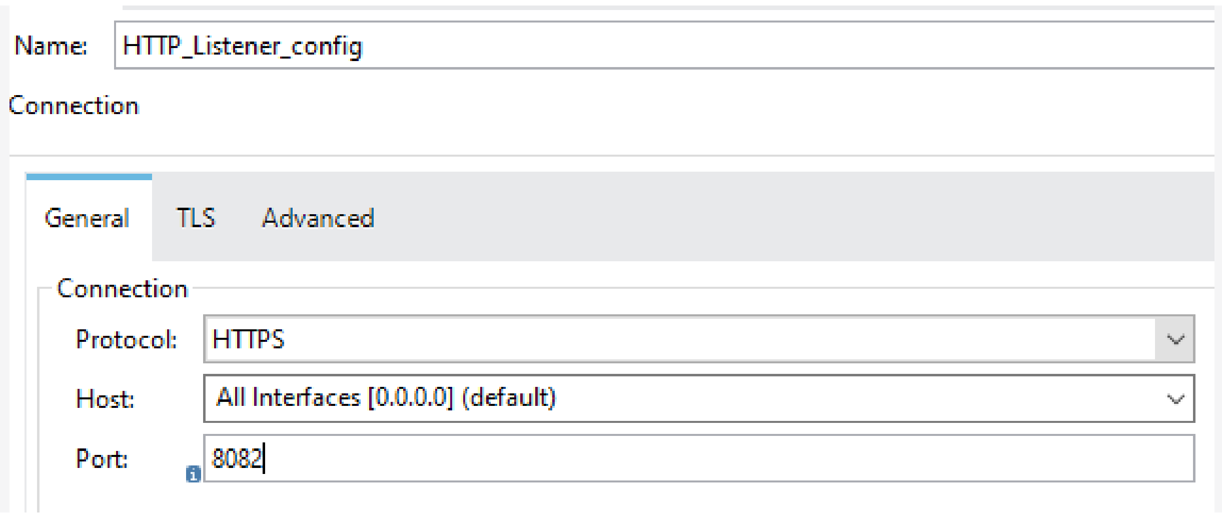
```
C:\Program Files\Java\jdk1.8.0_251\bin>keytool -import -alias mule-client-public -keystore C:/Certificates/client-truststore.jks -file C:/Certificates/server_public.cr

Enter keystore password:
Re-enter new password:
Owner: CN=localhost, OU=Information Technology, O=Self, L=Mumbai, ST=Maharashtra, C=In
Issuer: CN=localhost, OU=Information Technology, O=Self, L=Mumbai, ST=Maharashtra, C=In
Serial number: 3f08042d
Valid from: Tue Aug 18 18:25:56 IST 2020 until: Mon Nov 16 18:25:56 IST 2020
Certificate fingerprints:
         MD5:  44:EB:BE:7B:D4:9E:54:BD:8C:32:51:D1:A8:EA:8E:06
         SHA1: BB:9F:1E:98:96:BE:D5:DC:2E:7D:B2:E5:5B:DC:34:64:F4:9F:DA:AD
         SHA256: 7F:BF:A5:FE:FF:74:21:0C:7E:C4:57:F7:F3:DC:DF:C4:6A:29:90:73:CA:1A:46:04:E2:ED:22:F3:CA:DB:B8:7C
Signature algorithm name: SHA256withRSA
Subject Public Key Algorithm: 2048-bit RSA key
Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 63 59 DE 16 DA 96 24 EA   E9 3C 08 09 70 71 33 CD  cY....$..<..pq3.
0010: 01 15 2D 2F                                        ..-/
]
]

Trust this certificate? [no]:  yes
Certificate was added to keystore
```

Now, we have generated the server Keystore and client truststore.

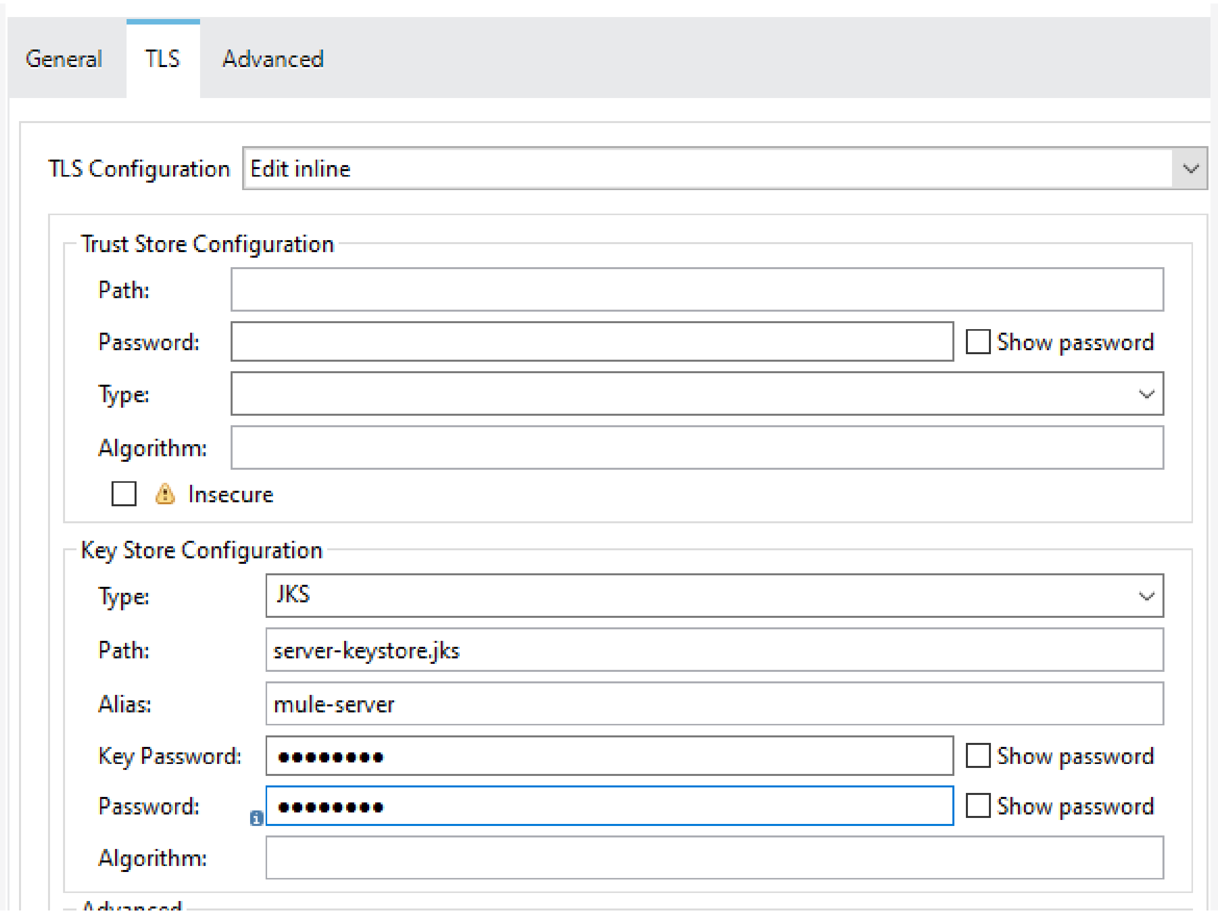
Step 4: Configuring MuleSoft HTTP Listener and Requester

Let's see how we can configure Keystore and truststore on the MuleSoft application.

Now, we will configure the server-Keystore on the MuleSoft HTTP Listener. Under General Settings, the Protocol must be "HTTPS". We need to make sure **server-Keystore.jks and client-truststore.jks** that are generated in the above steps, must be copied under folder **src/main/resources**.

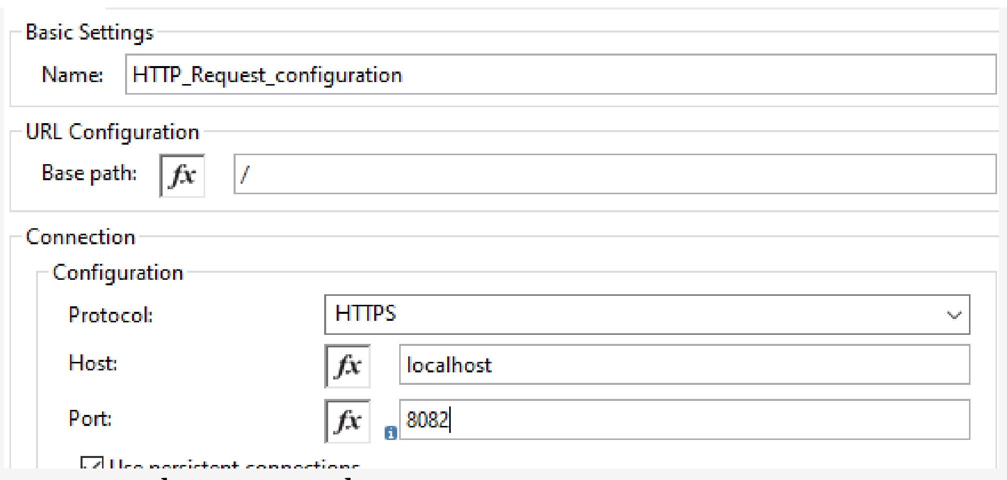

Name: HTTP_Listener_config
Connection
General | TLS | Advanced
Connection
Protocol: HTTPS
Host: All Interfaces [0.0.0.0] (default)
Port: 8082

Now, we will perform a TLS Key Store configuration. Provider Type, Keystore Path, Keystore and key password, and alias name.

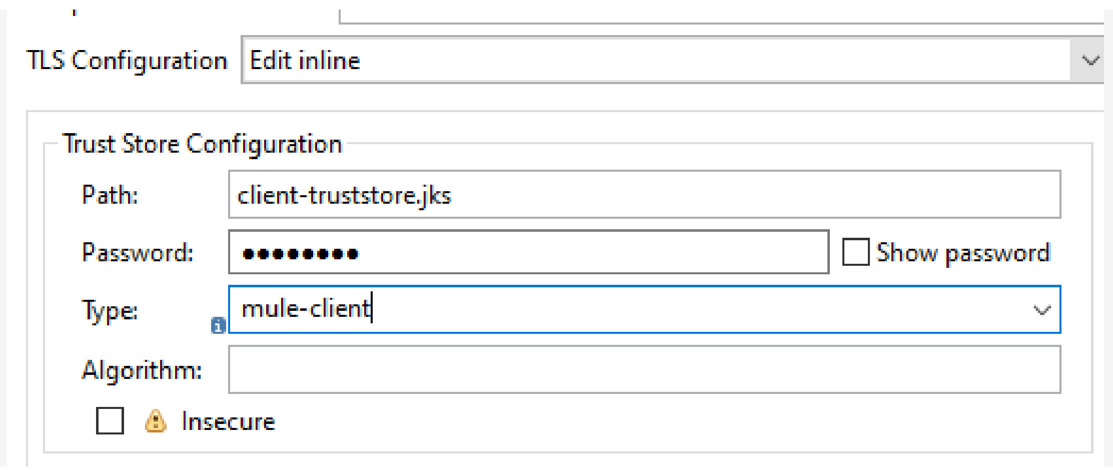

General | TLS | Advanced
TLS Configuration: Edit inline
Trust Store Configuration
Path:
Password: ☐ Show password
Type:
Algorithm:
☐ Insecure
Key Store Configuration
Type: JKS
Path: server-keystore.jks
Alias: mule-server
Key Password: •••••••• ☐ Show password
Password: •••••••• ☐ Show password
Algorithm:

This is the configuration that needs to be done on the server-side.

Now, we will see the client-side configuration. For that, we will be using the MuleSoft HTTP requester. Provide the connection settings on the HTTP requester.


Basic Settings
Name: HTTP_Request_configuration
URL Configuration
Base path: /
Connection
Configuration
Protocol: HTTPS
Host: localhost
Port: 8082

Now, we will do the TLS configuration. Provide the client's truststore path, password, type, etc.


TLS Configuration: Edit inline
Trust Store Configuration
Path: client-truststore.jks
Password: •••••••• ☐ Show password
Type: mule-client
Algorithm:
☐ Insecure

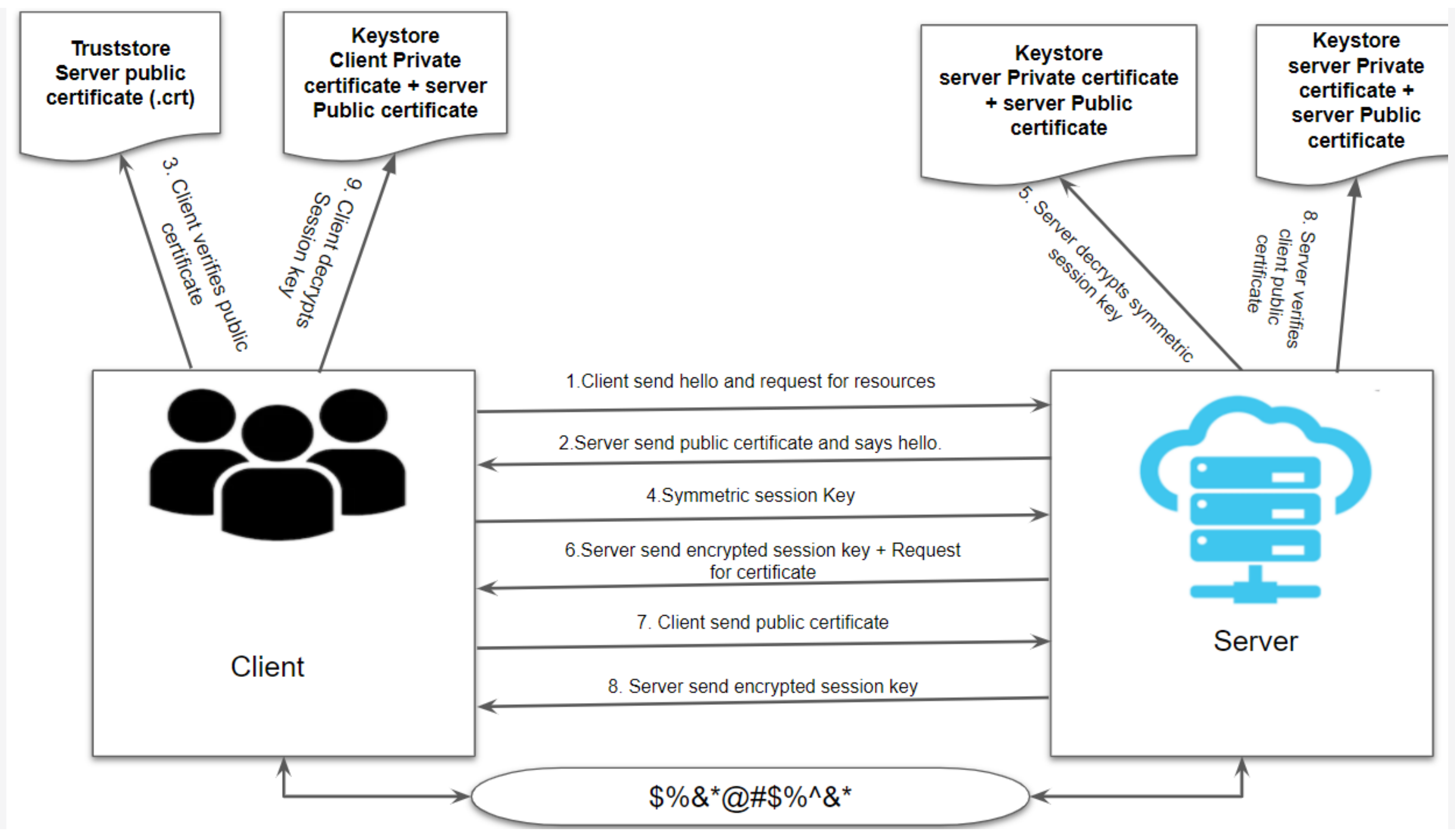
For more details on the implementing one-way SSL for the MuleSoft application, please go through below video tutorial.

Two Way SSL (Mutual Authentication)

As mentioned above in two ways SSL client verifies the server certificates and the server verifies the client certificates.

At the server end, there will be a Keystore which will hold the private and public certificate of the server and truststore which will hold the public certificate of client whereas, at the client end, there will be a Keystore which will hold the private and public certificate of client whereas truststore which will hold the public key of the server.

- Clients will send Hello and request for the resources on the secure HTTPS protocol.
- The server will respond with its public certificate (.crt) and send Hello.
- The client will verify the server public certificate in its truststore.
- The client sends back symmetric session key generated using the server public certificate.
- The server will decrypt the symmetric session key using the server private certificate and request for the client certificate.
- The client will send its public certificate to the server and the server will verify the client public certificate in the server truststore.
- The server will generate a session key and encrypt using the client public certificate and send it to the client.
- The client will decrypt the session key using client private certificate and this way the key exchange between client and server. It will establish secure communication between client and server.


Truststore Server public certificate (.crt)
Keystore Client Private certificate + server Public certificate
Keystore server Private certificate + server Public certificate
Keystore server Private certificate + server Public certificate
3. Client verifies public certificate
9. Client decrypts Session key
5. Server decrypts symmetric session key
8. Server verifies client public certificate
1.Client send hello and request for resources
2.Server send public certificate and says hello.
4.Symmetric session Key
6.Server send encrypted session key + Request for certificate
7. Client send public certificate
8. Server send encrypted session key
Client
Server
$%&*@#$%^&*

Implementing Two Way SSL For MuleSoft Application

For generating Keystore and truststore, we can use OpenSSL or keytool utility. For this article, we will be using keytool which is part of your JDK.

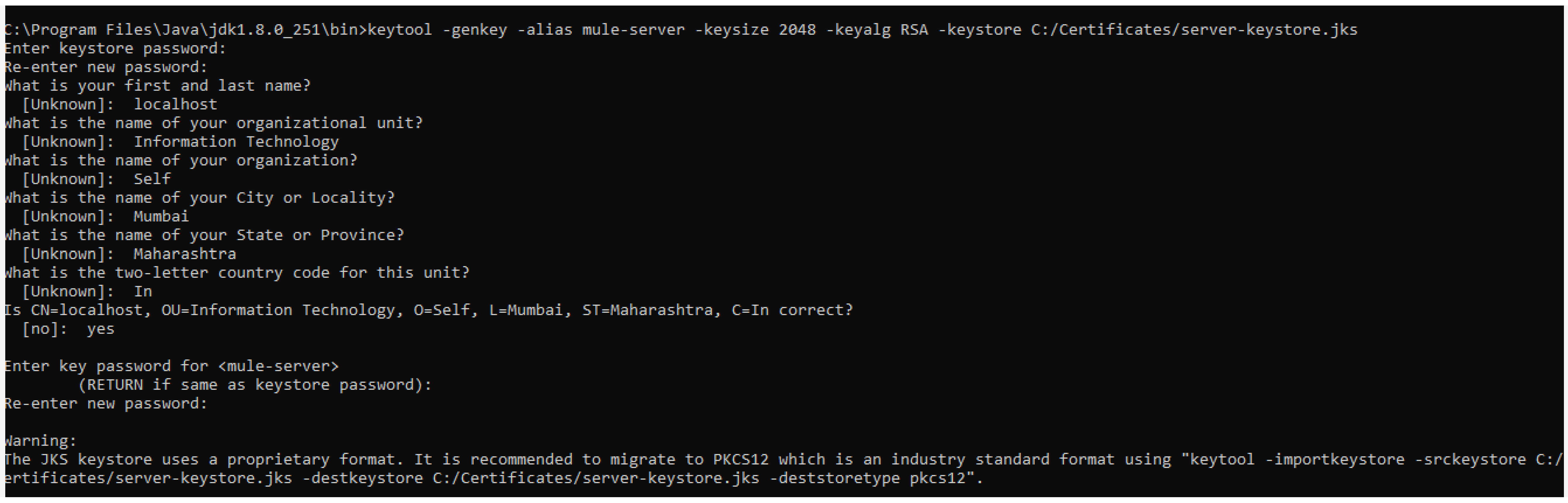
Step 1: Generate Server Keystore

For generating the server Keystore, we will be using the below command.

Plain Text
```
keytool -genkey -alias mule-server -keysize 2048 -keyalg RSA -keystore C:/Certificates/server-keystore.jks
```

Once we execute the above command, it will ask for the Keystore password, first and last name including other details and the key password.

alias in the above command is used to search the certificate in the Keystore as there can be multiple certificates in the same Keystore and alias is useful in identifying the right certificate.

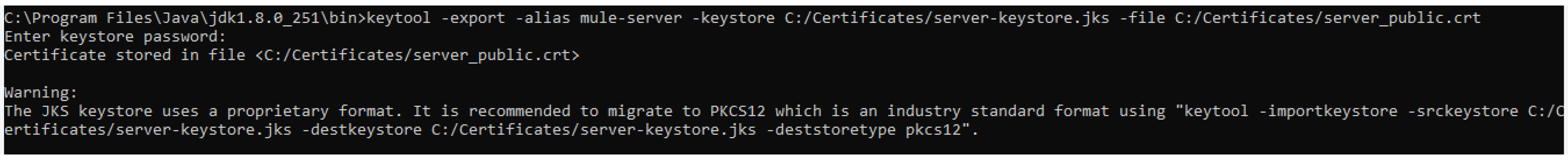

```
C:\Program Files\Java\jdk1.8.0_251\bin>keytool -genkey -alias mule-server -keysize 2048 -keyalg RSA -keystore C:/Certificates/server-keystore.jks
Enter keystore password:
Re-enter new password:
What is your first and last name?
  [Unknown]:  localhost
What is the name of your organizational unit?
  [Unknown]:  Information Technology
What is the name of your organization?
  [Unknown]:  Self
What is the name of your City or Locality?
  [Unknown]:  Mumbai
What is the name of your State or Province?
  [Unknown]:  Maharashtra
What is the two-letter country code for this unit?
  [Unknown]:  In
Is CN=localhost, OU=Information Technology, O=Self, L=Mumbai, ST=Maharashtra, C=In correct?
  [no]:  yes

Enter key password for <mule-server>
        (RETURN if same as keystore password):
Re-enter new password:

Warning:
The JKS keystore uses a proprietary format. It is recommended to migrate to PKCS12 which is an industry standard format using "keytool -importkeystore -srckeystore C:/
ertificates/server-keystore.jks -destkeystore C:/Certificates/server-keystore.jks -deststoretype pkcs12".
```

Step 2: Export the Public Certificate From Server Keystore

We will be requiring the server public certificate to be installed in the client truststore. So we need to extract the public key from the server Keystore. For extracting the server public certificate, we will be using the below export command.

Plain Text
```
keytool -export -alias mule-server -keystore C:/Certificates/server-keystore.jks -file C:/Certificates/server_public.crt
```

This command will export the server public certificate and we will import that server public certificate in the client truststore.

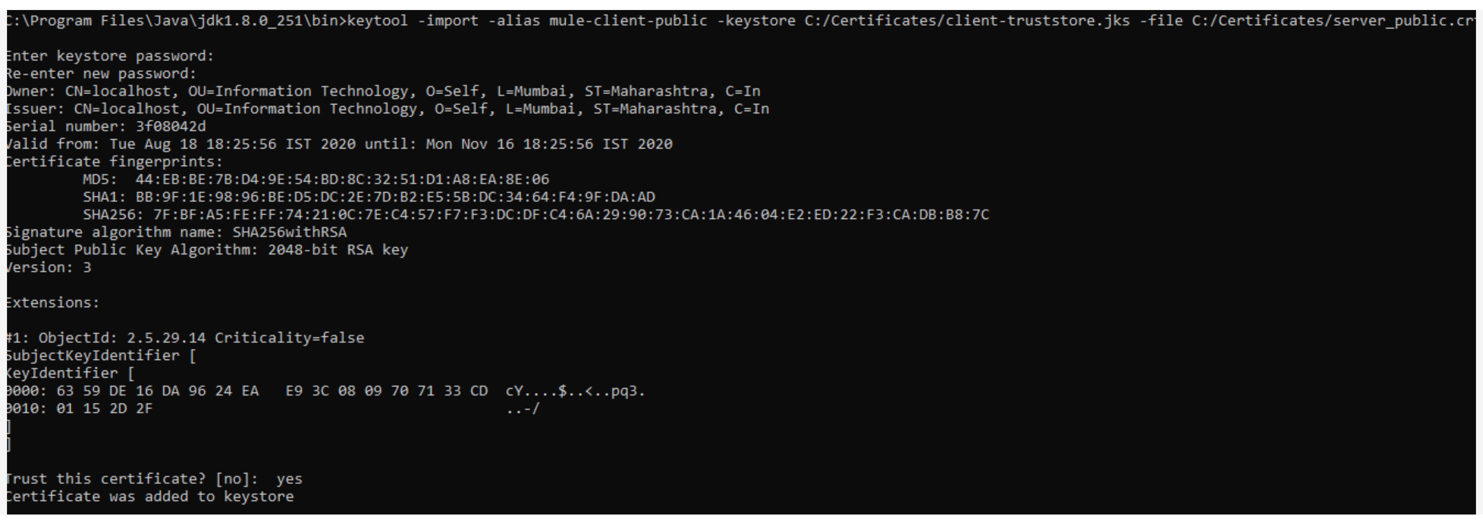

```
C:\Program Files\Java\jdk1.8.0_251\bin>keytool -export -alias mule-server -keystore C:/Certificates/server-keystore.jks -file C:/Certificates/server_public.crt
Enter keystore password:
Certificate stored in file <C:/Certificates/server_public.crt>

Warning:
The JKS keystore uses a proprietary format. It is recommended to migrate to PKCS12 which is an industry standard format using "keytool -importkeystore -srckeystore C:/C
ertificates/server-keystore.jks -destkeystore C:/Certificates/server-keystore.jks -deststoretype pkcs12".
```

Step 3: Import Server Public Certificate Into Client Truststore

For importing the server public key into the client truststore, we will be using the below command.

Plain Text
```
keytool -import -alias mule-client-public -keystore C:/Certificates/client-truststore.jks -file C:/Certificates/server_public.crt
```

This above command will generate the client's truststore and import the server public certificate.


```
C:\Program Files\Java\jdk1.8.0_251\bin>keytool -import -alias mule-client-public -keystore C:/Certificates/client-truststore.jks -file C:/Certificates/server_public.cr

Enter keystore password:
Re-enter new password:
Owner: CN=localhost, OU=Information Technology, O=Self, L=Mumbai, ST=Maharashtra, C=In
Issuer: CN=localhost, OU=Information Technology, O=Self, L=Mumbai, ST=Maharashtra, C=In
Serial number: 3f08042d
Valid from: Tue Aug 18 18:25:56 IST 2020 until: Mon Nov 16 18:25:56 IST 2020
Certificate fingerprints:
         MD5:  44:EB:BE:7B:D4:9E:54:BD:8C:32:51:D1:A8:EA:8E:06
         SHA1: BB:9F:1E:98:96:BE:D5:DC:2E:7D:B2:E5:5B:DC:34:64:F4:9F:DA:AD
         SHA256: 7F:BF:A5:FE:FF:74:21:0C:7E:C4:57:F7:F3:DC:DF:C4:6A:29:90:73:CA:1A:46:04:E2:ED:22:F3:CA:DB:B8:7C
Signature algorithm name: SHA256withRSA
Subject Public Key Algorithm: 2048-bit RSA key
Version: 3

Extensions:

#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 63 59 DE 16 DA 96 24 EA   E9 3C 08 09 70 71 33 CD  cY....$..<..pq3.
0010: 01 15 2D 2F                                        ..-/
]
]

Trust this certificate? [no]:  yes
Certificate was added to keystore
```

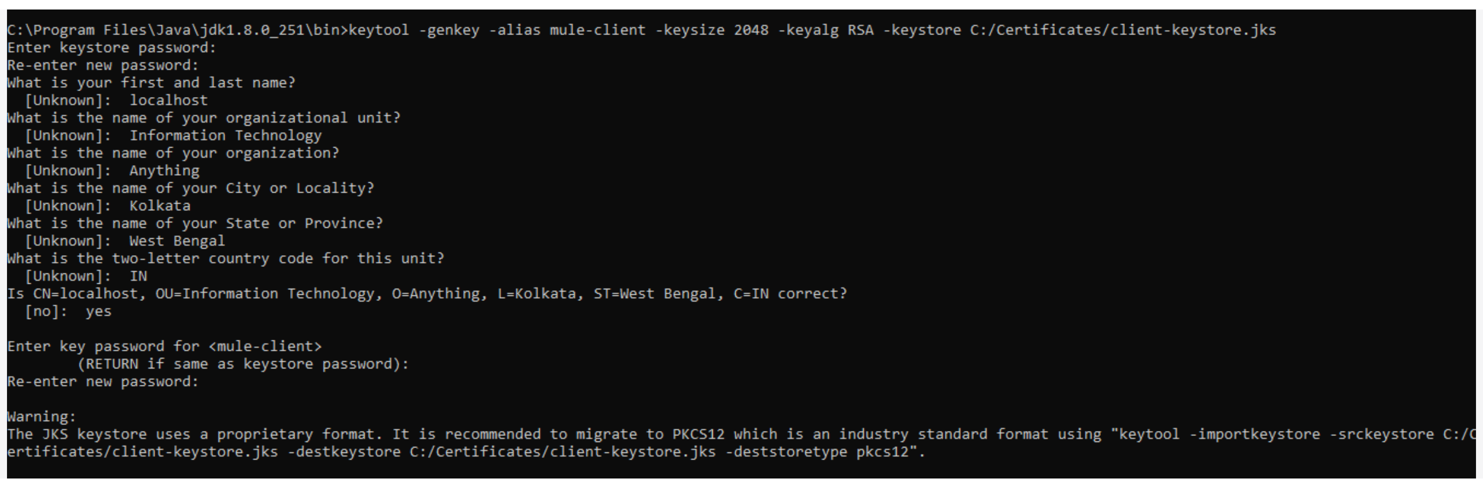
Step 4: Generate Client Keystore

For generating the client Keystore, we will be using the below command.

Plain Text
```
keytool -genkey -alias mule-client -keysize 2048 -keyalg RSA -keystore C:/Certificates/client-keystore.jks
```

Once we execute the above command, it will ask for the Keystore password, first and last name including other details and the key password.

Alias in the above command is used to search the certificate in the Keystore as there can be multiple certificates in the same Keystore and alias is useful in identifying the right certificate.



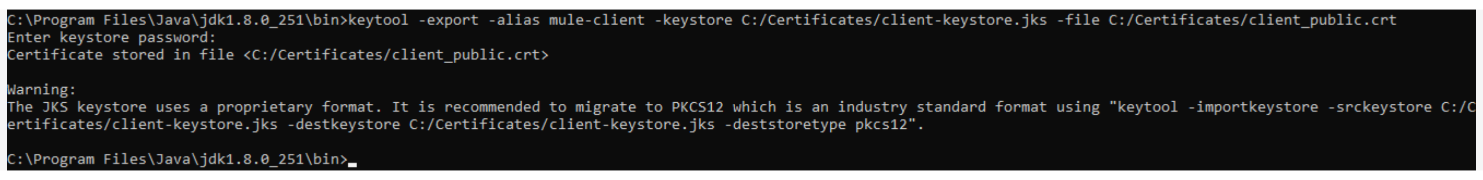
```
C:\Program Files\Java\jdk1.8.0_251\bin>keytool -genkey -alias mule-client -keysize 2048 -keyalg RSA -keystore C:/Certificates/client-keystore.jks
Enter keystore password:
Re-enter new password:
What is your first and last name?
  [Unknown]:  localhost
What is the name of your organizational unit?
  [Unknown]:  Information Technology
What is the name of your organization?
  [Unknown]:  Anything
What is the name of your City or Locality?
  [Unknown]:  Kolkata
What is the name of your State or Province?
  [Unknown]:  West Bengal
What is the two-letter country code for this unit?
  [Unknown]:  IN
Is CN=localhost, OU=Information Technology, O=Anything, L=Kolkata, ST=West Bengal, C=IN correct?
  [no]:  yes

Enter key password for <mule-client>
        (RETURN if same as keystore password):
Re-enter new password:

Warning:
The JKS keystore uses a proprietary format. It is recommended to migrate to PKCS12 which is an industry standard format using "keytool -importkeystore -srckeystore C:/C
ertificates/client-keystore.jks -destkeystore C:/Certificates/client-keystore.jks -deststoretype pkcs12".
```

Step 5: Export the Public Certificate From Client Keystore

We will be requiring the client public certificate to be installed in the server truststore. So we need to extract the public key from the client Keystore. For extracting the client public certificate, we will be using the below export command.

Plain Text
```
keytool -export -alias mule-client -keystore C:/Certificates/client-keystore.jks -file C:/Certificates/client_public.crt
```

This command will export the client public certificate and we will import that client public certificate in the server truststore.



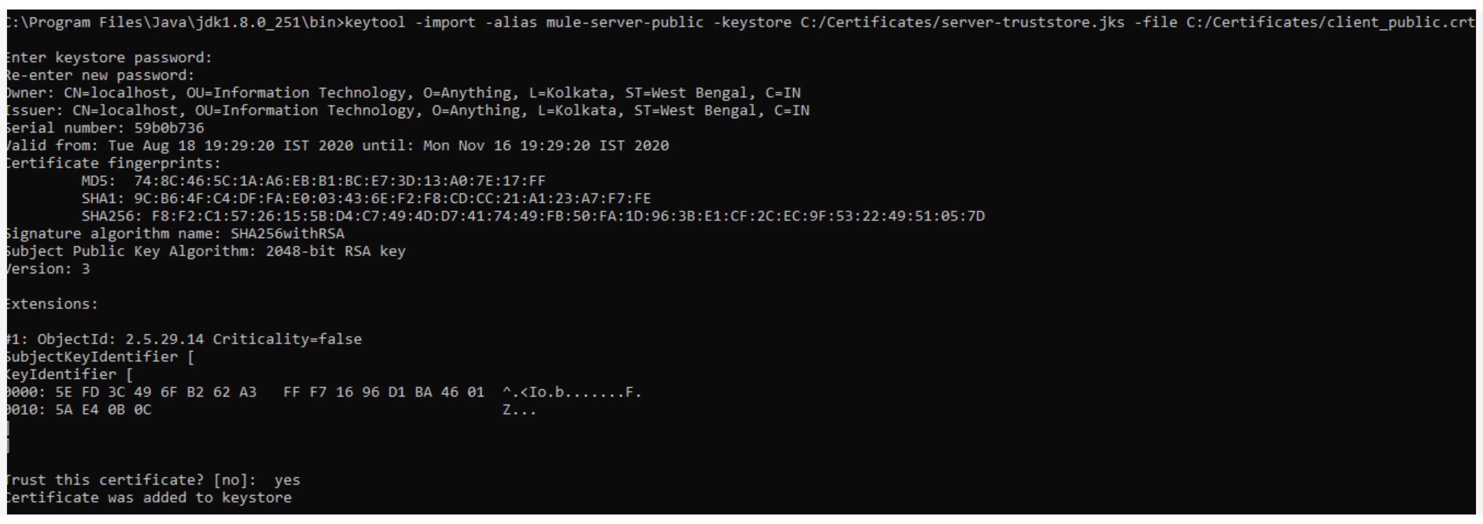
```
C:\Program Files\Java\jdk1.8.0_251\bin>keytool -export -alias mule-client -keystore C:/Certificates/client-keystore.jks -file C:/Certificates/client_public.crt
Enter keystore password:
Certificate stored in file <C:/Certificates/client_public.crt>

Warning:
The JKS keystore uses a proprietary format. It is recommended to migrate to PKCS12 which is an industry standard format using "keytool -importkeystore -srckeystore C:/C
ertificates/client-keystore.jks -destkeystore C:/Certificates/client-keystore.jks -deststoretype pkcs12".

C:\Program Files\Java\jdk1.8.0_251\bin>
```

Step 6: Import Client Public Certificate Into Server truststore

For importing the client public certificate into the server truststore, we will be using the below command.

Plain Text
```
keytool -import -alias mule-server-public -keystore C:/Certificates/server-truststore.jks -file C:/Certificates/client_public.crt
```

This above command will generate the server truststore and import the client public certificate.



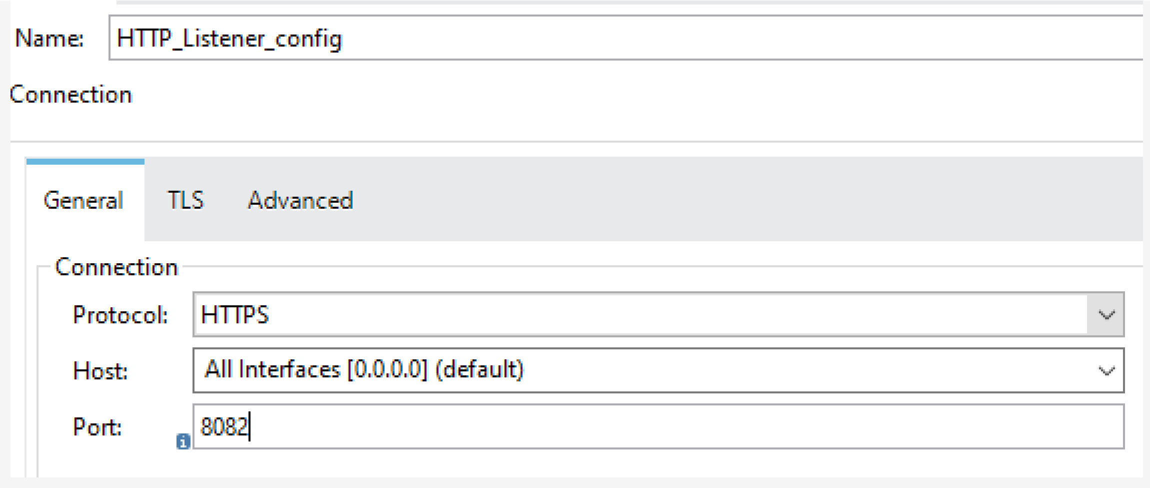
```
:\Program Files\Java\jdk1.8.0_251\bin>keytool -import -alias mule-server-public -keystore C:/Certificates/server-truststore.jks -file C:/Certificates/client_public.crt

nter keystore password:
e-enter new password:
wner: CN=localhost, OU=Information Technology, O=Anything, L=Kolkata, ST=West Bengal, C=IN
ssuer: CN=localhost, OU=Information Technology, O=Anything, L=Kolkata, ST=West Bengal, C=IN
erial number: 59b0b736
alid from: Tue Aug 18 19:29:20 IST 2020 until: Mon Nov 16 19:29:20 IST 2020
ertificate fingerprints:
         MD5:  74:8C:46:5C:1A:A6:EB:B1:BC:E7:3D:13:A0:7E:17:FF
         SHA1: 9C:B6:4F:C4:DF:FA:E0:03:43:6E:F2:F8:CD:CC:21:A1:23:A7:F7:FE
         SHA256: F8:F2:C1:57:26:15:5B:D4:C7:49:4D:D7:41:74:49:FB:50:FA:1D:96:3B:E1:CF:2C:EC:9F:53:22:49:51:05:7D
ignature algorithm name: SHA256withRSA
ubject Public Key Algorithm: 2048-bit RSA key
ersion: 3

xtensions:

1: ObjectId: 2.5.29.14 Criticality=false
ubjectKeyIdentifier [
eyIdentifier [
000: 5E FD 3C 49 6F B2 62 A3   FF F7 16 96 D1 BA 46 01  ^.<Io.b.......F.
010: 5A E4 0B 0C                                        Z...


rust this certificate? [no]:  yes
ertificate was added to keystore
```

Now, we have generated the server Keystore, server truststore, client Keystore and client truststore.

Step 7: Configuring MuleSoft HTTP Listener and Requester

Let's see how we can configure Keystore and truststore on the MuleSoft application.

Now, we will configure server-Keystore and client-truststore on the MuleSoft HTTP Listener. Under General Settings, the Protocol must be "HTTPS". We need to make sure of the **server-Keystore.jks, server-truststore.jks, client-Keystore.jks, and client-truststore.jks**that are generated in the above steps, must be copied under folder **src/main/resources**.

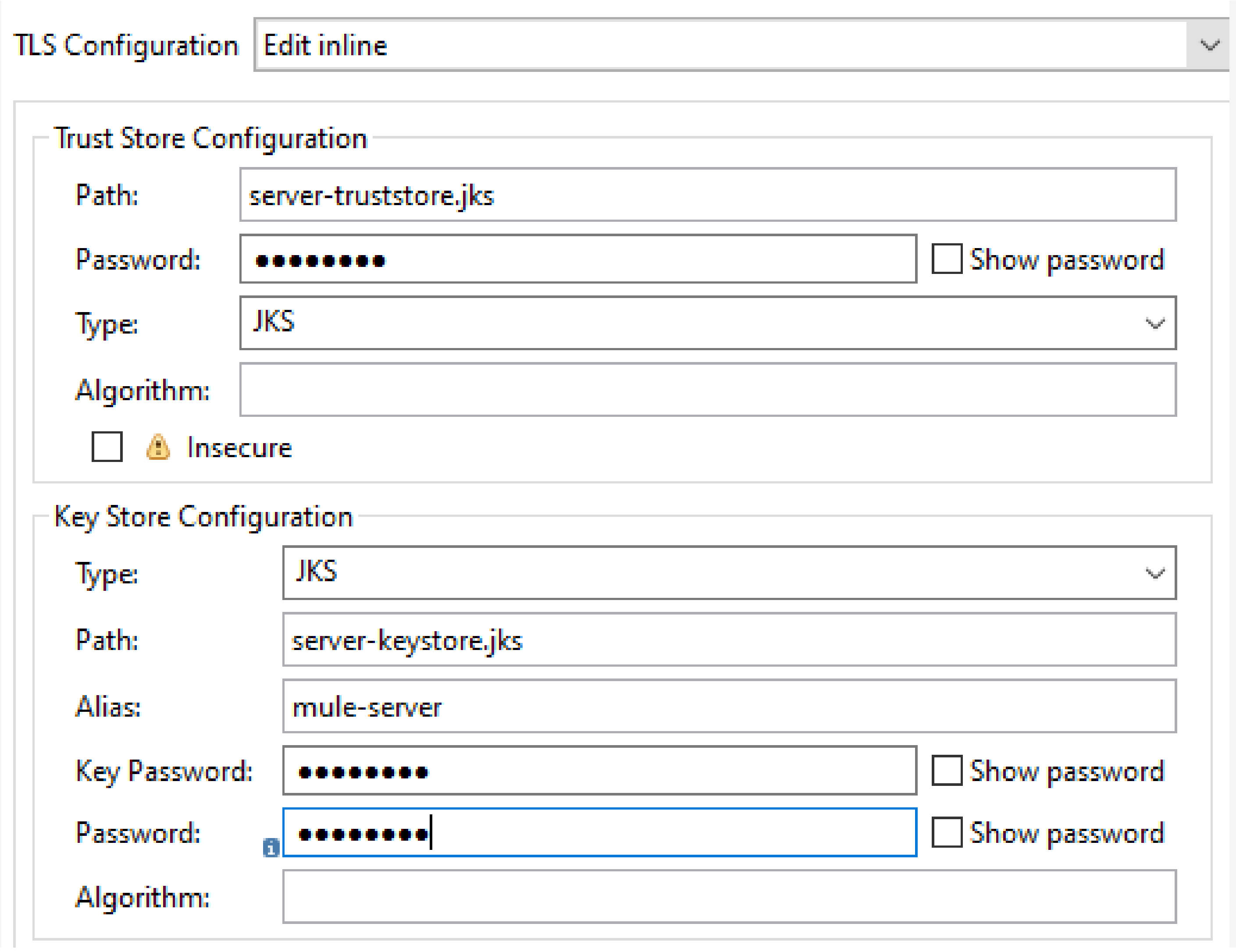


Name: HTTP_Listener_config

Connection

Connection
Protocol: HTTPS
Host: All Interfaces [0.0.0.0] (default)
Port: 8082

Now, we will perform TLS Key Store. Provide Type, Keystore Path, Keystore, and key password and alias name. Perform the Trust Store Configuration. Provide Path, Password, and Type.



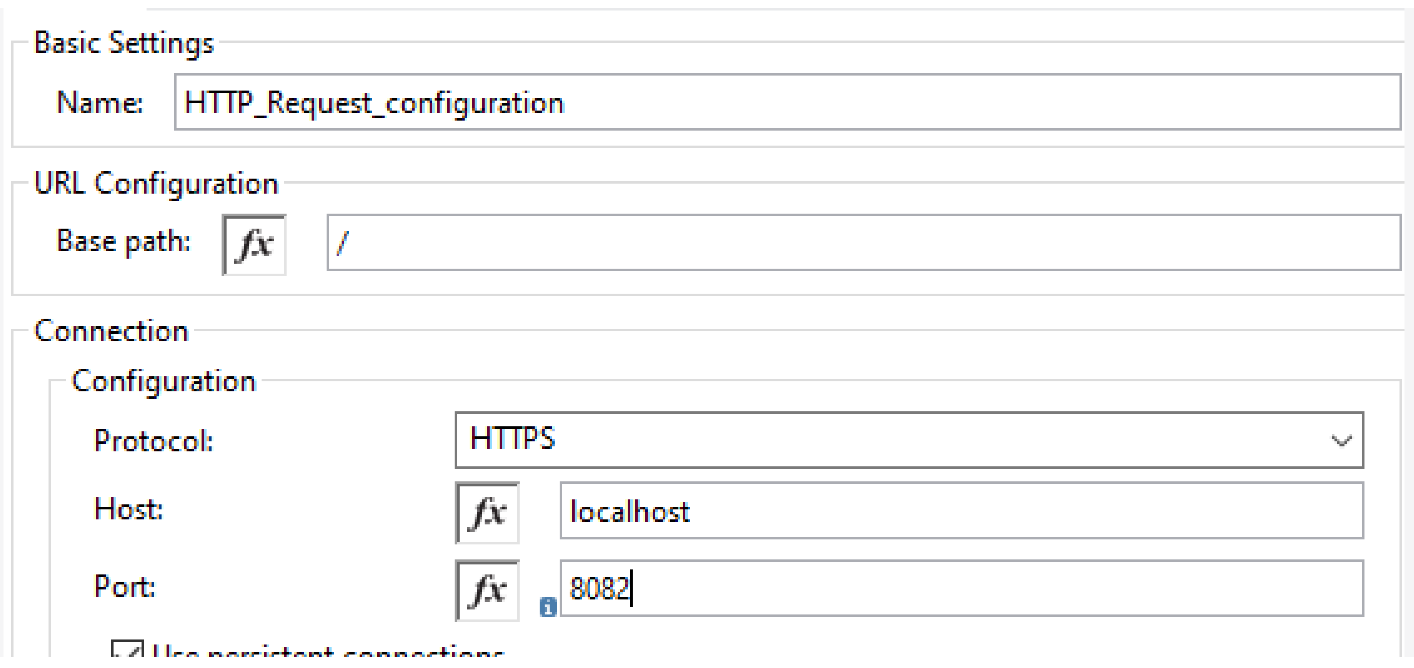
TLS Configuration: Edit inline

Trust Store Configuration
Path: server-truststore.jks
Password: ••••••••
Type: JKS
Algorithm:
☐ Insecure

Key Store Configuration
Type: JKS
Path: server-keystore.jks
Alias: mule-server
Key Password: ••••••••
Password: ••••••••
Algorithm:

This is the configuration that needs to be done on the server-side.

Now, we will see the client-side configuration. For that, we will be using the MuleSoft HTTP requester. Provide the connection settings on the HTTP requester.

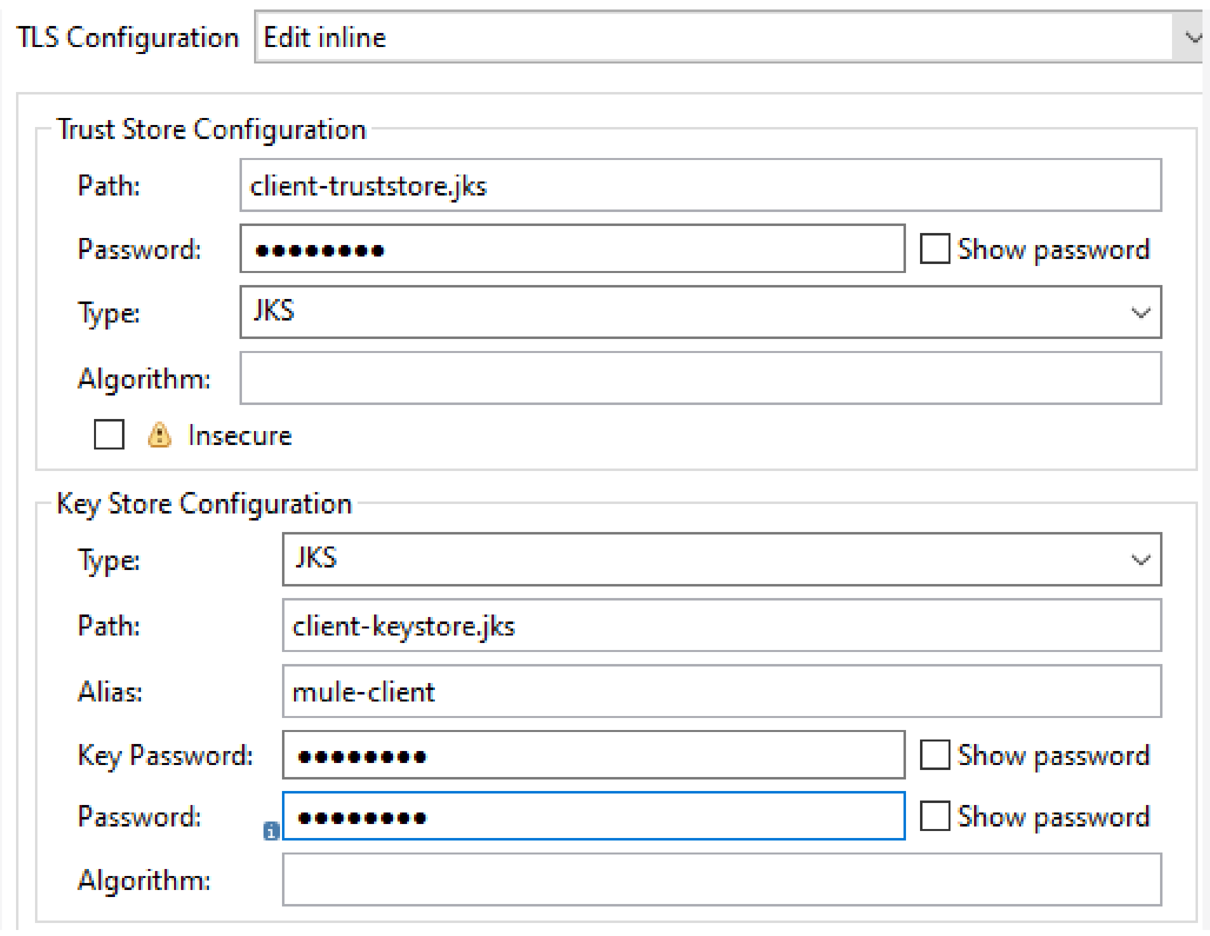


Basic Settings
Name: HTTP_Request_configuration

URL Configuration
Base path: /

Connection
Configuration
Protocol: HTTPS
Host: localhost
Port: 8082

Now, we will do the TLS configuration. Provide the client Trust Store Path, Password, and Type. Provide the Key Store Path, Password, Type, and Path.



TLS Configuration: Edit inline

Trust Store Configuration
Path: client-truststore.jks
Password: ●●●●●●●● ☐ Show password
Type: JKS
Algorithm:
☐ Insecure

Key Store Configuration
Type: JKS
Path: client-keystore.jks
Alias: mule-client
Key Password: ●●●●●●●● ☐ Show password
Password: ●●●●●●●● ☐ Show password
Algorithm:

For more details on the implementing two-way SSL for the MuleSoft application, please go through below video tutorial.

Now, you know how to implement one way and two way SSL for MuleSoft applications.