**Step-by-Step Implementation**

**1. Generate Server Key Pair and Keystore**

**Command:**

keytool -genkeypair -alias serverKey -keyalg RSA -keystore serverKeystore.jks -keysize 2048

**Explanation:**

* keytool: The Java Key and Certificate Management Tool.
* -genkeypair: Generates a key pair (public and private keys).
* -alias serverKey: The alias for the entry in the keystore; you can choose any name, but it should be unique within the keystore.
* -keyalg RSA: The algorithm to be used for generating the key pair (RSA is a common choice).
* -keystore serverKeystore.jks: The name of the keystore file where the generated key pair will be stored. You can change this as needed.
* -keysize 2048: The size of the key in bits; 2048 is a standard size providing good security.

**2. Export Server Certificate**

**Command:**

keytool -export -alias serverKey -keystore serverKeystore.jks -file serverCert.cer

**Explanation:**

* -export: Exports the certificate from the keystore.
* -alias serverKey: The alias for the entry you want to export; should match the alias used when generating the key pair.
* -keystore serverKeystore.jks: The keystore file where the key pair is stored.
* -file serverCert.cer: The name of the file to save the exported certificate.

**3. Create Client Truststore and Import Server Certificate**

**Command:**

keytool -import -alias serverCert -file serverCert.cer -keystore clientTruststore.jks

**Explanation:**

* -import: Imports a certificate into a truststore.
* -alias serverCert: The alias for the certificate you are importing; it should be unique within the truststore.
* -file serverCert.cer: The certificate file you exported previously.
* -keystore clientTruststore.jks: The name of the truststore file where the certificate will be stored.

**4. Generate Client Key Pair and Keystore**

**Command:**

keytool -genkeypair -alias clientKey -keyalg RSA -keystore clientKeystore.jks -keysize 2048

**Explanation:**

* Similar to the server keystore generation command.

**5. Export Client Certificate**

**Command:**

keytool -export -alias clientKey -keystore clientKeystore.jks -file clientCert.cer

**Explanation:**

* This exports the client certificate from the keystore.

**6. Create Server Truststore and Import Client Certificate**

**Command:**

keytool -import -alias clientCert -file clientCert.cer -keystore serverTruststore.jks

**Explanation:**

* This imports the client certificate into the server's truststore.

**Summary of Changes**

* **Aliases**: Choose meaningful and unique aliases for each entry.
* **Keystore/Truststore Names**: Update filenames and paths according to your directory structure and naming conventions.
* **File Names**: Ensure exported certificate files have the desired names and paths.

By following these steps, you can set up secure two-way SSL communication between the client and server. This ensures that both parties can verify each other's identities, enhancing the security of your application

**Steps to Generate Server and Client Certificates and Truststores in pks**

**1. Generate Server Key Pair and Keystore**

bash

Copy code

keytool -genkeypair -alias serverKey -keyalg RSA -keystore serverKeystore.p12 -storetype PKCS12 -keysize 2048

**Output:** serverKeystore.p12

**2. Export Server Certificate**

bash

Copy code

keytool -export -alias serverKey -keystore serverKeystore.p12 -storetype PKCS12 -file serverCert.cer

**Output:** serverCert.cer

**3. Generate Client Key Pair and Keystore**

bash

Copy code

keytool -genkeypair -alias clientKey -keyalg RSA -keystore clientKeystore.p12 -storetype PKCS12 -keysize 2048

**Output:** clientKeystore.p12

**4. Export Client Certificate**

bash

Copy code

keytool -export -alias clientKey -keystore clientKeystore.p12 -storetype PKCS12 -file clientCert.cer

**Output:** clientCert.cer

**5. Create Truststores**

**Client Truststore (import server certificate):**

bash

Copy code

keytool -import -alias serverCert -file serverCert.cer -keystore clientTruststore.p12 -storetype PKCS12

**Server Truststore (import client certificate):**

bash

Copy code

keytool -import -alias clientCert -file clientCert.cer -keystore serverTruststore.p12 -storetype PKCS12

**6. Optional: Convert PKCS#12 to PEM Format (if needed for cURL)**

**Convert Server Keystore to PEM:**

bash

Copy code

openssl pkcs12 -in serverKeystore.p12 -out serverKeystore.pem -nodes -nocerts

**Convert Client Keystore to PEM:**

bash

Copy code

openssl pkcs12 -in clientKeystore.p12 -out clientKeystore.pem -nodes -nocerts

**7. Convert Certificates to PEM Format (if needed)**

bash

Copy code

openssl x509 -in serverCert.cer -out serverCert.pem -outform PEM

openssl x509 -in clientCert.cer -out clientCert.pem -outform PEM

**8. Testing with cURL**

Once you have the PEM files, you can test your API using cURL as follows:

bash

Copy code

curl -X POST https://localhost:8443/financial \

-d '{"key": "value"}' \

-H "Content-Type: application/json" \

--cert clientKeystore.pem \

--key clientKeystore.pem \

--cacert serverCert.pem