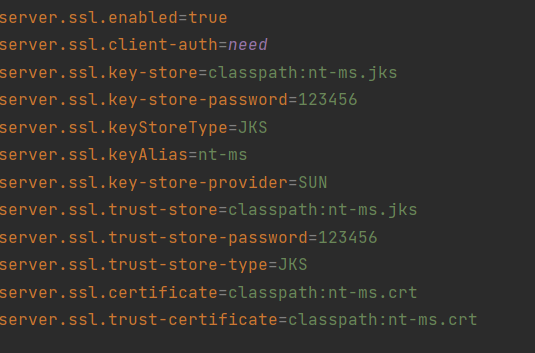
Data Transfer with SSL

1. Use **KEYTOOL** to generate **PKC12 (.p12)** file which holds keys for **server**, do same for **client A** and **client B**
2. Create certificate .**crt** file from that **PKC12** for each server.
3. Import **server's crt** into **Client A's** **.p12 file** and **vise versa**
4. Configure and **enable SSL** in application properties in each server.



*Note: configure client-auth as* ***need*** *for* ***server*** *and as* ***none*** *for* ***client****.*

1. For client-side, **load** the .p12 file in **Keystore** to create **SslContext.**
2. Use that sslContext in HttpClient for secure connection.

That's all for configuration.

1. When client A hits any endpoint of server, client checks if the server's public key is Trusted or not and vice versa, if it's not trusted then it throws an error with bad credentials else it gives access to it.

***Note: Both server's and client's public key needs to be in each other's trustedCred in .p12 in order to execute successful HandShake***

**Command to generate .p12 and crt file**

SOME KEYTOOL COMMAND

Create keystore and generate keypair :

keytool -genkeypair -alias <my\_keypair\_alias> -keyalg RSA -keysize 2048 -storetype PKCS12 -keystore <my\_store\_name>.p12 -validity 3650

Create public certificate file :

keytool -export -alias <my\_keypair\_alias> -file <my\_cert\_name>.crt -keystore <my\_store\_name>.p12

Import Client Cert to Server jks File

keytool -import -alias <my\_keypair\_alias -file <my\_cert\_name>.crt -keystore <my\_store\_name>.p12

View certificate

keytool -list -keystore <my\_store\_name>.p12

keytool -list -v -keystore <my\_store\_name>.p12