

MATS Client Certificate Guide For Web Client & API App

EPEX SPOT
12.09.2025

Content

1.	Introduction	4
1.1	Audience	4
1.2	Purpose	4
1.3	Document revision history	4
2.	Certification rules and process	5
2.1	General information	5
2.2	Client certificate generation process overview	5
2.3	Technical information	7
2.4	CSR Creation Rules	7
2.5	Revocation of certificate	9
2.6	Expiration and Renewal of a certificate	10
2.7	Guidance about MATS SIMU Client Certificate	10
2.8	Guidance about MATS PROD Client Certificate	11
3.	JKS or PFX – Applicable for all customers	12
3.1	Mandatory creation of Key Store	12
3.2	JKS AND PKCS12 (PFX) Key Store File formats	12
3.3	Using JKS or PKCS12 (PFX) files	13

4.	MATS Web Trading Client Users	14
4.1	How to install your certificate? – Personal & Trusted Root Certificate Authorities	14
4.2	How to verify if you imported your certificate in your web browser?	18
5.	MATS API App Users	23
5.1	Using client certificate in an API app	23
5.2	Using JKS or PKCS12 with SoapUI &Postman	23
6.	Annex - Technical Guidance	24
6.1	Examples of CSR and KeyStore files generation	24
6.2	Troubleshooting	26

1. Introduction

1.1 Audience

This document is intended for customers who will use the MATS (API & Web Trading Client), EPEX SPOT Market Operators and EPEX SPOT IT Team.

1.2 Purpose

This documentation provides information about certificates needed to connect to MATS API Server.

It provides the technical information about certificates, the certificate management process and the process to obtain a certificate.

1.3 Document revision history

Evolution	Date	Version	Author	Comments
Creation	05.05.2025	1.0	EPEX SPOT API Customer Services	Creation
Update	13.05.2025	1.1	EPEX SPOT API Customer Services	Update the CSR creation rules. CN convention update. Include guidance about MATS SIMU & PROD Client Certificate Update technical information
Update	16.07.2025	1.2	EPEX SPOT API Customer Services	Update the certificate installation process. Updated the certificate guidance with explicit clarification about per Company == per MEMBER SHORT NAME as defined at the exchange
Update	09.09.2025	1.3	EPEX SPOT API Customer Services	Include the key usage and extended key usage attribute values for CSR creation rules Include openssl command line to generate PKCS12 file for Mac OS users

2. Certification rules and process

2.1 General information

The client certificate is mandatory for the MATS is a signed public key:

- generated by a trusted Certificate Authority (CA),
- based on a certificate signing request customers send to EPEX,
- which is created by the customer using the private key

A client certificate is per company. However, a customer can request for multiple Client Certificates per Company (if necessary) to fulfil the organizational arrangements.

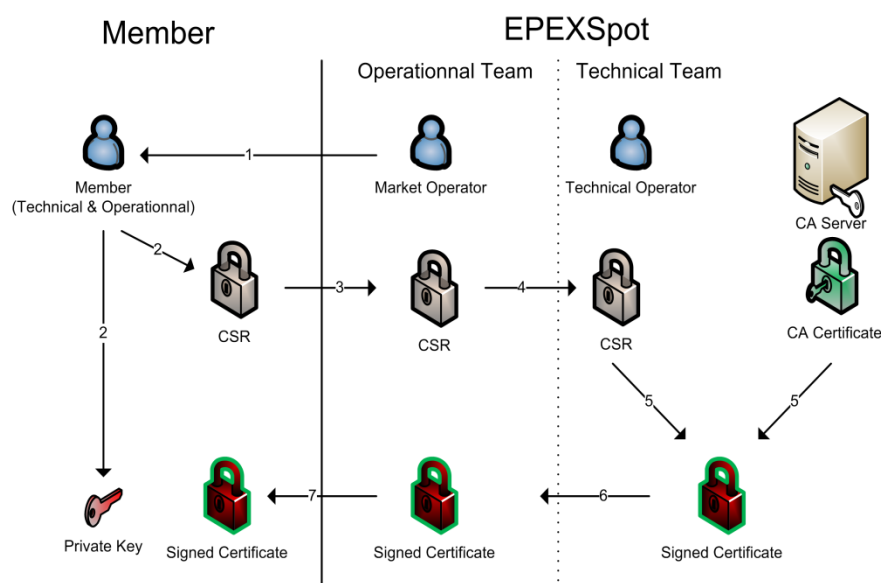
All customers who has Web trading client user accounts and/or API user accounts must generate the private key. Once the private key has been generated it can be used to generate the "Certificate Signing Request" file (CSR), as explained below.

Your private key should never be provided to anyone. Should for any reason the private part of the certificate be shared outside of the member, EPEX SPOT will not be able to guarantee the member identity.

The below sections will guide you through the required steps to obtain a signed certificate from EPEX and generate the mandatory technical file (keystore) that needs to establish a secure connection with MATS. Refer **section 3 & section 4** depending on which type of user account you use to connect to MATS.

2.2 Client certificate generation process overview

The following schema describes the certification process for customers (Members and Market Data Customers):



Step	Description
1	Market Operators validate the Member/Market Data customer identity (referred below as the “customer”).
2	The customer first generates the private key , and once the key has been generated the CSR can be generated using that private key.
3	The customer sends by email the CSR file to Market Operators but does NOT share the private key .
4	Market Operators transfer the CSR file to a Technical Operator in order to get the certificate signed.
5	The Technical Operator uses the CSR file and the Certificate Authority (CA) certificate to generate the Signed certificate (.pem file).
6	The Technical Operator transfers the signed certificate (.pem file) to Market Operators.
7	Market Operators send by email the signed certificate (.pem file) to the customer.
8	What to do with the signed certificate (.pem file)? : please refer to section 3 & section 4 explaining: ▪ How to install certificate for the MATS trading web client usage? ▪ How to create keystore and use

Notes:

- The exchanged files (CSR File and Signed Certificate file) are public and can be exchanged by email.
- You can double check your CSR content by using a CSR decoder such as <https://certlogik.com/decoder/>
- The only valid certificates that can be used with the MATS API will be signed by the EPEX Certificate Authority.

2.3 Technical information

In order to ensure a secure communication with MATS, the following solutions have been implemented:

- All communications between MATS Clients (Web Trading Client & API apps) and MATS server are encrypted, using HTTPS
- Bi-lateral authentication system which requires a Client certificate to connect

2.3.1 Supported TLS version

The following TLS versions are supported by MATS:

- Protocol TLS 1.2
- We will stick to TLS v1.2 until further notice.

2.3.2 Supported list of cipher suites

List of supported cipher suites:

Cipher suites
TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256
TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384
TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384

2.3.3 Supported key & signature algorithms

- **ECC¹** with:
 - Key size ≥ **256** bits
 - All signature algorithms supported by ECC
- **RSA** with:
 - Key size ≥ **4096** bits
 - The recommended signature algorithm for RSA is **sha256WithRSAEncryption**.

2.4 CSR Creation Rules

Each certificate is unique and is identified by a combination of Country Name / Organization Name / Common Name.

¹ Elliptic Curve Cryptography (ECC)

This information is used by EPEX SPOT IT for the validation of the CSR and the generation of the signed certificate.

The following conditions must be met for the creation of the CSR:

- **Only ASCII characters are accepted**
- **Country Name** (2 letter code) :
 - o Must respect the country of the company (Validated by Operators)
- **Organization Name:**
 - o Must meet your **company short name**
 - o Please contact **Market Operation** if you do not remember your company short name, or for API clients use the “name” in the response to a *Retrieve Participant Info* request (cf. MATS REST API portal)
- **Common Name:**
 - o your *Organization Name* + “_”+**your Exchange** + “_”+ **environment**
 - where **your Exchange** should be *EPEX* or *SVPX*
 - use **EPEX** if you are a member at EPEX exchange
 - use **SVPX** if you are a member at any of the Serviced Exchange(s)
 - advice is to use **EPEX** if you are a member at both EPEX and any of the Serviced Exchange(s)
 - o **Remark:** In such case, if member use EPEX or SVPX does not have technical or functional impact. Both values are acceptable. The above point is just “clear advice” to avoid any confusions.
 - where **environment** is **Simu** or **Prod**
 - use **Simu** if the certificate is required for MATS SIMU environment(s)
 - use **Prod** if the certificate is required for MATS PROD environment
 - Example:
 - For test environments SIMU1 and SIMU2: *COMPANY_EPEX_Simu*
 - For Production: *COMPANY_EPEX_Prod*
 - o must be an UTF8String (can be verified in ASN.1 Information of the CSR)
 - o When creating CSR, ensure the values X509v3 Key Usage & X509v3 Extended Key Usage as indicated below

X509v3 Key Usage: critical

Digital Signature, Key Encipherment

X509v3 Extended Key Usage:

TLS Web Client Authentication

If you have several certificates the content of all of the certificates (Common_Name, Organization Name and Country) are exactly the same.

2.5 Revocation of certificate

In case the member would like to revoke a certificate (e.g. if the private key got exposed), the process is as follows:

- The member contacts Market Operators, who validate the member identity
- The member provides the **Country Name / Organization Name / Common Name** combination which identifies the certificate to be revoked
- The member is contacted (by email) by Market Operators to confirm the revocation of his certificate. Once revoked, a certificate cannot be used anymore.

2.6 Expiration and Renewal of a certificate

The certificates are valid for 2 years in PRODUCTION and 2 years in SIMULATION.

EPEX Market operators monitor PRODUCTION certificates expiry dates and informs customers one month before the expiry date.

The validity period of a certificate cannot be extended but, as long as the private key has not changed, the same CSR can be re-used and signed again, leading to a new signed certificate (new .pem file) distributed to customers.

If the private key has changed, then the corresponding new CSR must be provided to EPEX Market Operators.

Note: The same “**Country Name / Organization Name / Common Name**” combination can be used for the new CSR.

2.7 Guidance about MATS SIMU Client Certificate

No	Guideline
1	Client certificate is issued <u>per Company**</u>. This means all users (API users & Web client users) from the same company can use 1 certificate to login to MATS. **Note: Here <u>per Company</u> means per MEMBER SHORT NAME as defined at the exchange. In the external world, even if there is only one company, it is possible you have more than company accounts at the exchange under different SHORT NAMES. If this the case then, you need certificate per MEMBER SHORT NAME as defined at the exchange.
2	If a company is active in EPEX exchange and in one or several Serviced Exchanges(s)², then you need only one client certificate applicable for both EPEX & Serviced exchange(s). Optionally, customer can request for dedicated MATS client certificates for EPEX & SVPX (if necessary). Please follow the CSR process described in section 2.4 if you need dedicated MATS client certificate(s).
3	Existing ETS API customers who already has ETS API Simu Certificate can reuse the same certificate in MATS SIMU environment. This is applicable for both EPEX & Serviced Exchange(s).
4	Optionally, customer can request for dedicated MATS client certificates (if necessary)³.

² Serviced Exchanges are: SEMOPX, HUPX, BSP, SEEPEX, MEMO, MEPX

³ This could be due to several reasons:

- 1) Customer prefers to have dedicated client certificate for API app and Trading Desks who access MATS Web Trading Client.
- 2) Customer prefers to have dedicated client certificate per API app if there are more than one API apps exists within a company and/or managed by different departments

Please follow the CSR process described in **section 2.4** if you need dedicated MATS client certificate(s).

2.8 Guidance about MATS PROD Client Certificate

Kindly note that in contrary to MATS SIMU, ETS API PROD Certificate cannot be used for MATS PROD.

No	Guideline
1	Client certificate is issued <u>per Company</u>**. This means all users (API users & Web client users) from the same company can use 1 certificate to login to MATS. **Note: Here <u>per Company</u> means per MEMBER SHORT NAME as defined at the exchange. In the external world, even if there is only one company, it is possible you have more than company accounts at the exchange under different SHORT NAMES. If this the case then, you need certificate per MEMBER SHORT NAME as defined at the exchange.
2	If a company is active in EPEX exchange and in one or several Serviced Exchanges(s)⁴, then you need only one client certificate applicable for both EPEX & Serviced exchange(s). Optionally, customer can request for dedicated MATS client certificates for EPEX & SVPX (if necessary). Please follow the CSR process described in <u>section 2.4</u> if you need dedicated MATS client certificate(s).
3	Optionally, customer can request for dedicated MATS client certificates (if necessary)⁵. Please follow the CSR process described in <u>section 2.4</u> if you need dedicated MATS client certificate(s).

⁴ Serviced Exchanges are: SEMOPX, HUPX, BSP, SEEPEX, MEMO, MEPX

⁵ This could be due to several reasons:

- 3) Customer prefers to have dedicated client certificate for API app and Trading Desks who access MATS Web Trading Client.
- 4) Customer prefers to have dedicated client certificate per API app if there are more than one API apps exists within a company and/or managed by different departments

3. JKS or PFX – Applicable for all customers

Once you received the signed *.pem file, the IT contact from your company should follow the below steps which then allows you to perform the steps described in one of the below sections:

- **Section 4:** *For MATS Web Trading Client Users*
- **Section 5:** *For MATS API Apps*

3.1 Mandatory creation of Key Store

Your API application cannot directly use the signed certificate sent by EPEX. It needs in addition your certificate private key (generated at the same time as your CSR), combined in an appropriate container called a key store.

Key Store = your signed certificate + its private key

This KeyStore file (with any file extension) is required when the application wants to communicate over TLS through a secure channel.

The most popular keyStore files are:

- JKS (*Java KeyStore*), a Java proprietary format;
- PKCS12, one of the *Public-Key Cryptography Standards*, not Java specific

3.2 JKS AND PKCS12 (PFX) Key Store File formats

The biggest difference between **JKS** and **PKCS12** is that:

- **JKS** is a **format specific to Java** which stores private keys and certificates.
- **PKCS12** is a standardized and language-neutral way of storing encrypted private keys and certificates.

3.2.1 JKS Key store file format

The default and most widely used format for these files is JKS (Java Keystore) for a Java based application until Java 8. With Java 9, the default Keystore format changed from JKS to PKCS12.

That is until Java 8 your keystore format will be JKS if you don't specify the -storetype while creating your keystore with the *keytool* command. However, the default keystore type will be changed to PKCS12 in Java 9 because of its enhanced compatibility compared to JKS.

3.2.2 PKCS12 Key store file format

PKCS#12 is a file format (often called .p12 or .pfx) where you can store a private key and certificates. It's used for converting/transporting keys and certificates.

PKCS12, is a standard keystore type is not Java specific. It is portable and can be operated with libraries written in languages such as Java, C, C++ or C#.

3.3 Using JKS or PKCS12 (PFX) files

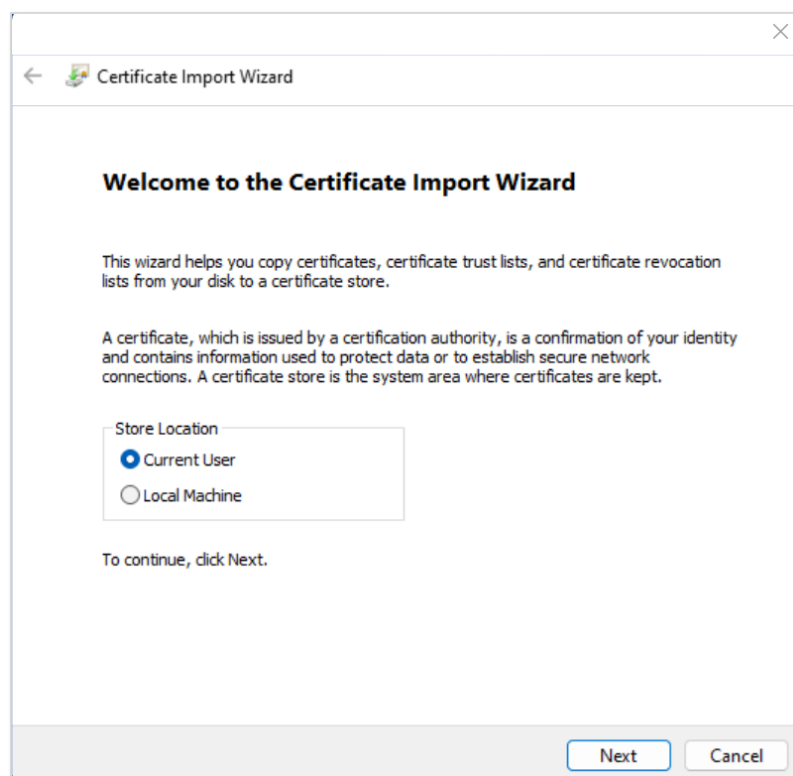
The decision on what Keystore type to use should be based on how you plan to use the private key-that is: what applications and/or web browsers will need to use the private key and what format(s) of key store do they already handle.

4. MATS Web Trading Client Users

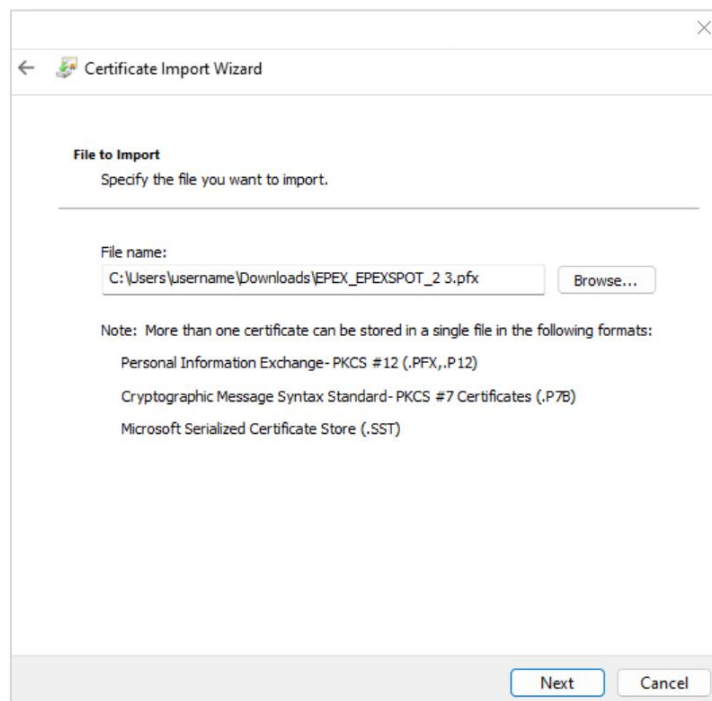
Once you received your signed certificate from EPEX (.pem file) you need to follow the below steps to get ready:

4.1 How to install your certificate? – Personal & Trusted Root Certificate Authorities

- EPEX signed the certificate and sent you a .pem file.
- **Step 1: Generate a .pfx file** out of the signed certificate (.pem file) sent by EPEX.
*Note: Please contact the relevant team/person from IT department within your company who handles such request for the necessary generation of *.pfx file.*
- **Step 2: Close all the browsers opened in your local machine and double click on the pfx file.** A Certificate Import Wizard will be displayed as shown below. Click on “Next” button:

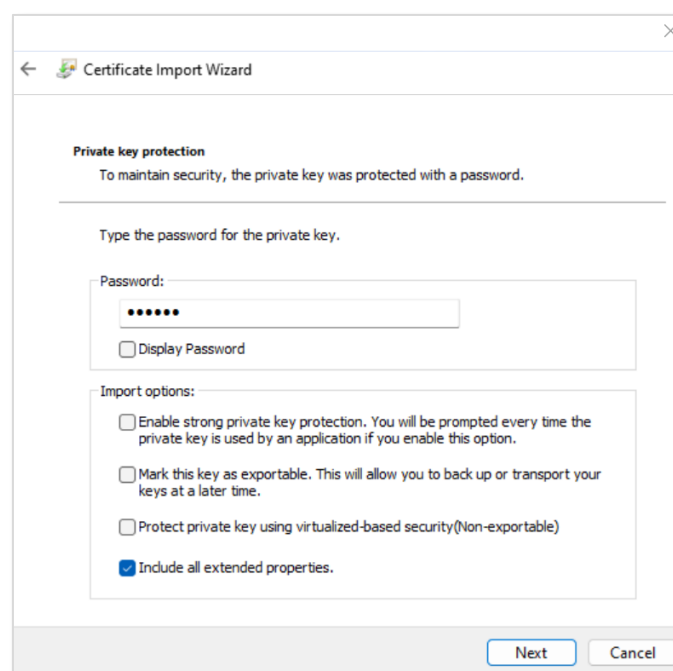


- **Step 3: Ensure the preselected File name is correct** and click on “Next” button.

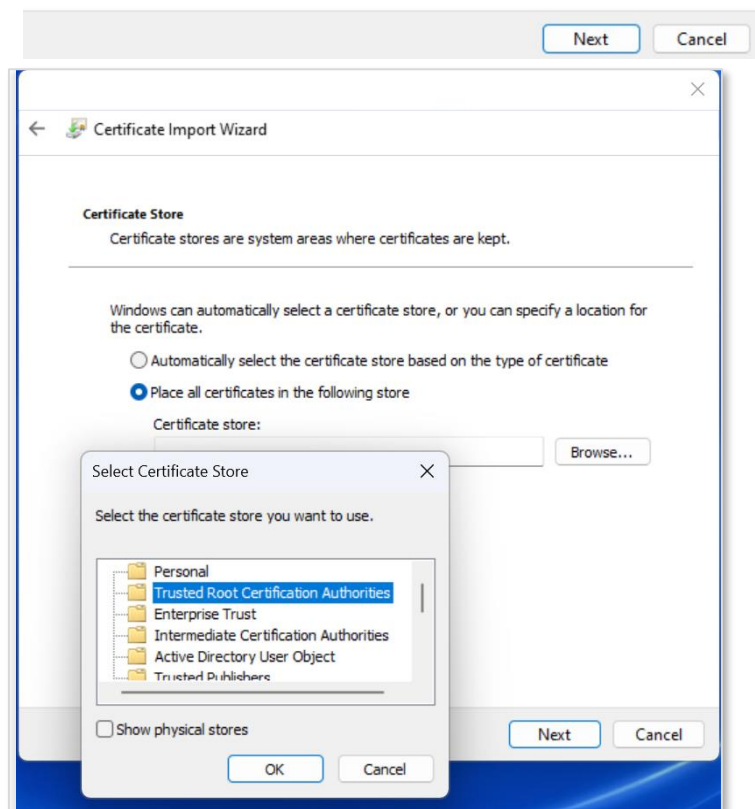
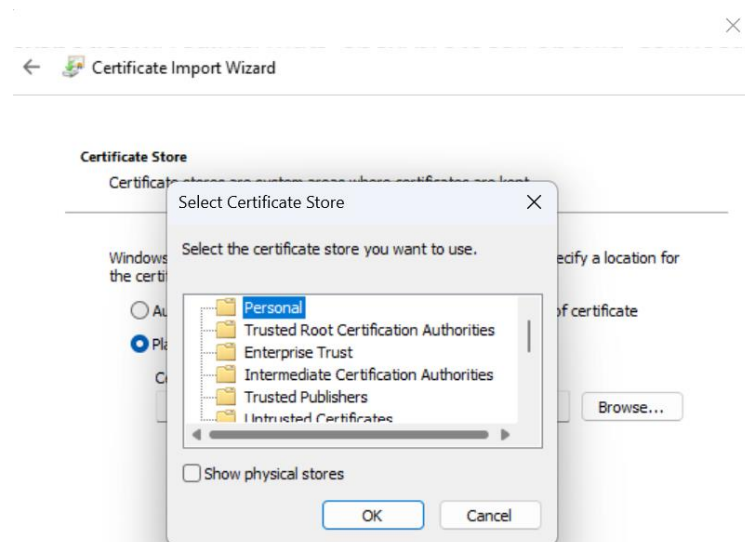


- **Step 4: If there is a password, then please enter the password.** Click on “Next”.

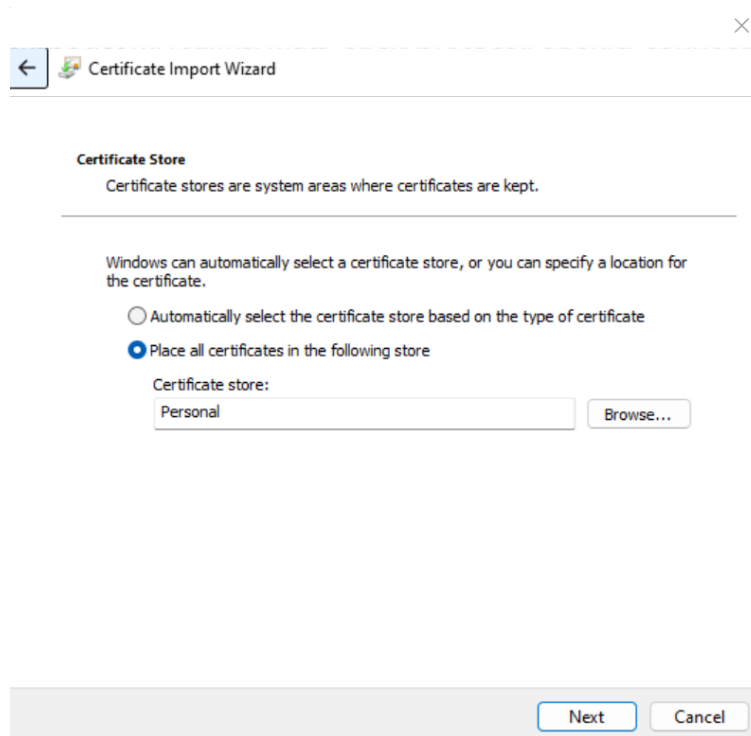
*Note: The password is set by the team/person who generated the *.pfx file at your company. Please contact the relevant person/team for this information.*



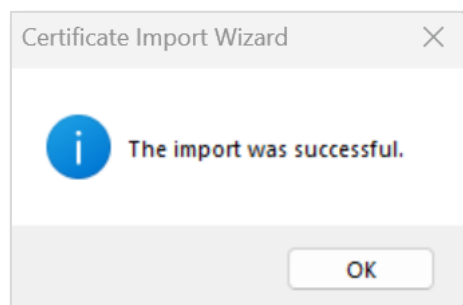
- **Step 5:** Select “Place all certificates in the following store” and click on “Browse”.
- **Step 6:** Select the “Personal” and click on “OK”.



- **Step 7:** Click on “Next”.



- **Step 8:** Click on “Finish”:
- **Step 9:** A successful message is displayed.

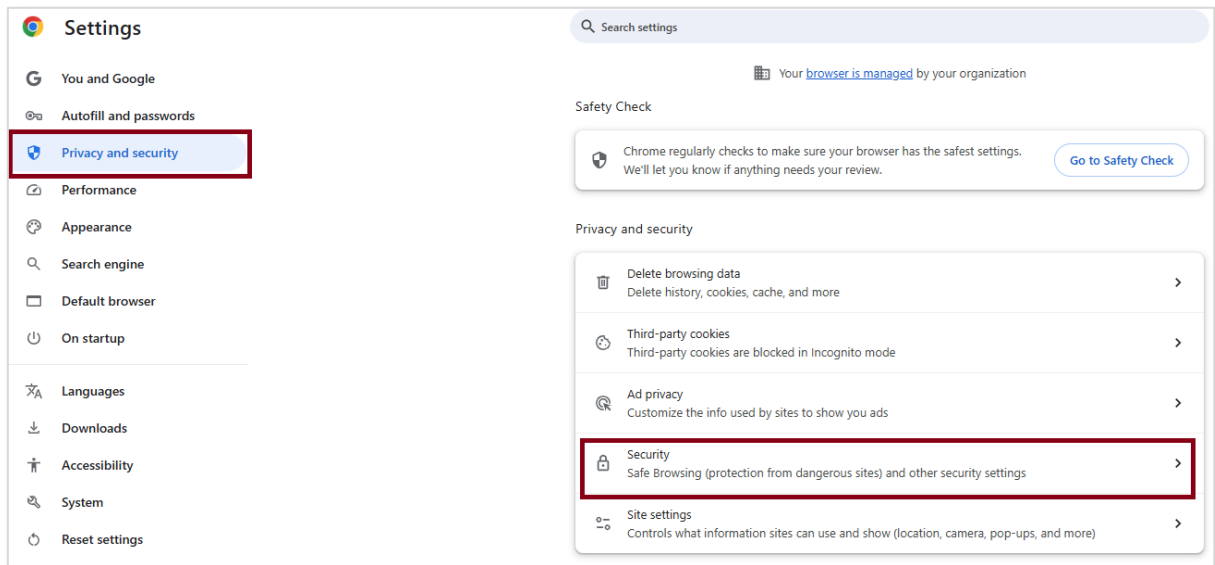


- **Step 10:** Repeat all the described above steps from **Step 2** until **Step 9**, by installing the certificate in “**Trusted Root Certificate Authorities**” folder

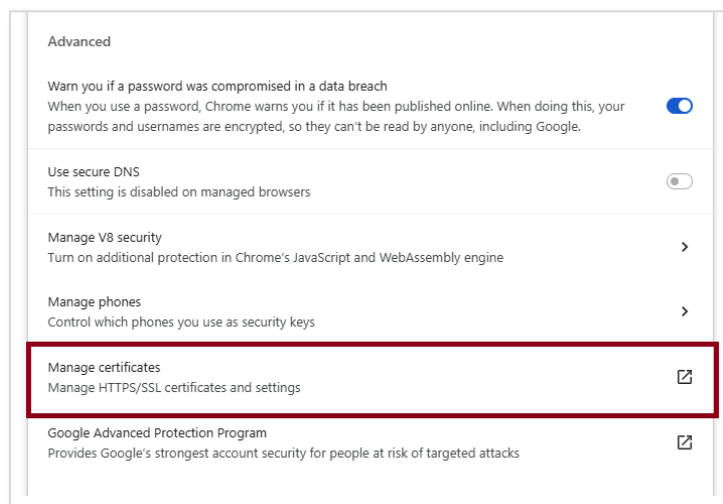
4.2 How to verify if you imported your certificate in your web browser?

4.2.1 Chrome

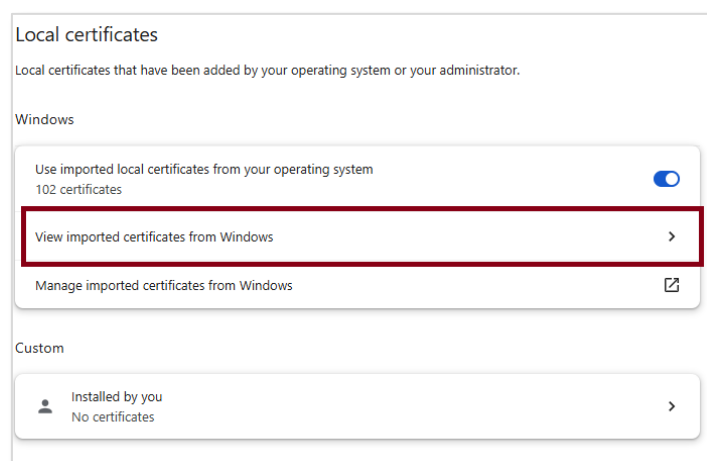
- Open Chrome → click on  (visible at the right top corner)→ click on “Settings”
- Click on “Privacy and Security”→click on “Security”



- Under the “Advanced” heading, click on “Manage certificates”



- Click on **“View imported certificates from windows”** under **“Local certificates”**



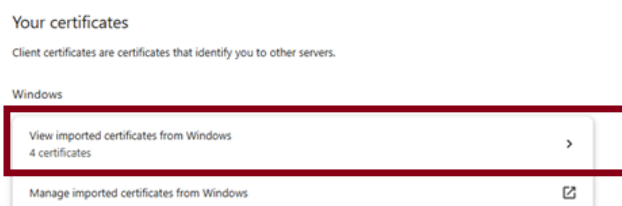
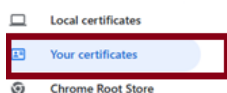
- Search for the **“Common Name (CN)”** that you provided within your CSR. If the import was successful, you will be able to find the certificate with the CN as shown below:

COMPANYA

4253c95af2dbc4647ec729ff16fc3eb5de...



- Repeat the verification by navigating to **“View imported certificates from windows”** under **“Your certificates”**



- Search for the **“Common Name (CN)”** that you provided within your CSR. If the import was successful, you will be able to find the certificate with the CN as shown below:

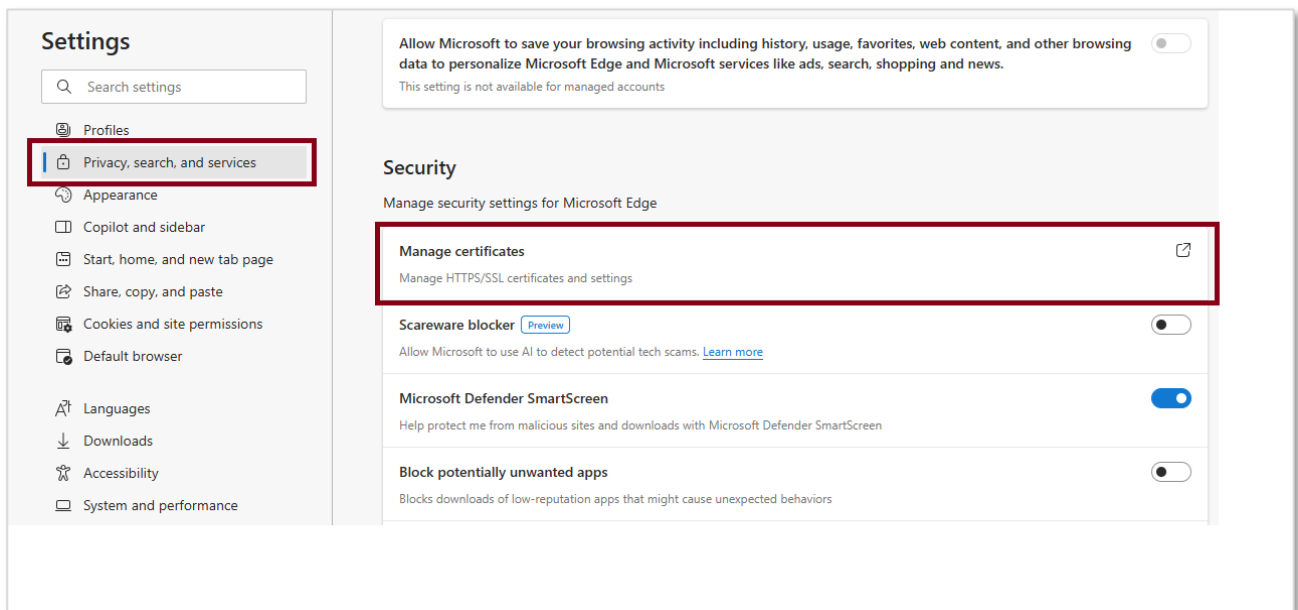
COMPANYA

4253c95af2dbc4647ec729ff16fc3eb5de...

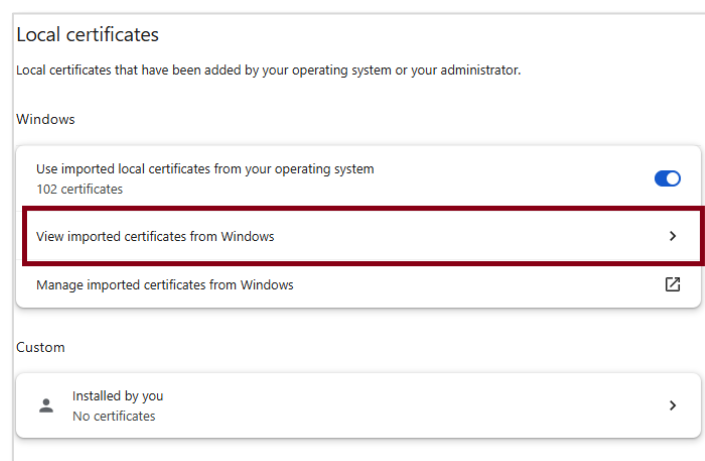


4.2.2 Microsoft Edge

- Open Microsoft Edge → click on ... (visible at the right top corner)→ click on “Settings”
- Click on “Privacy, search, and services”→click on “Manage certificates”



- Click on “View imported certificates from windows” under “Local certificates”



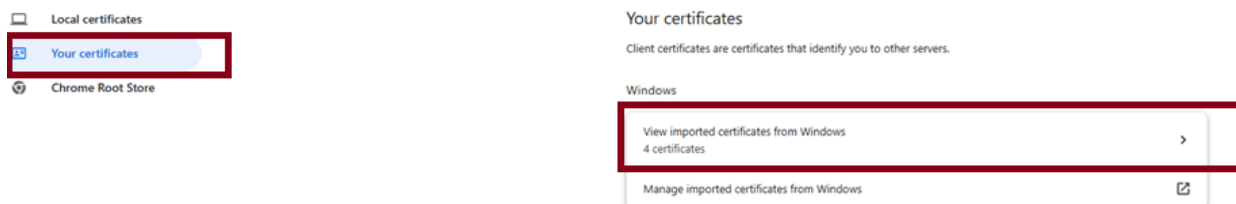
- Search for the “Common Name (CN)” that you provided within your CSR. If the import was successful, you will be able to find the certificate with the CN as shown below:

COMPANYA

4253c95af2dbc4647ec729ff16fc3eb5de...



- Repeat the verification by navigating to “**View imported certificates from windows**” under “**Your certificates**”



- Search for the “Common Name (CN)” that you provided within your CSR. If the import was successful, you will be able to find the certificate with the CN as shown below:

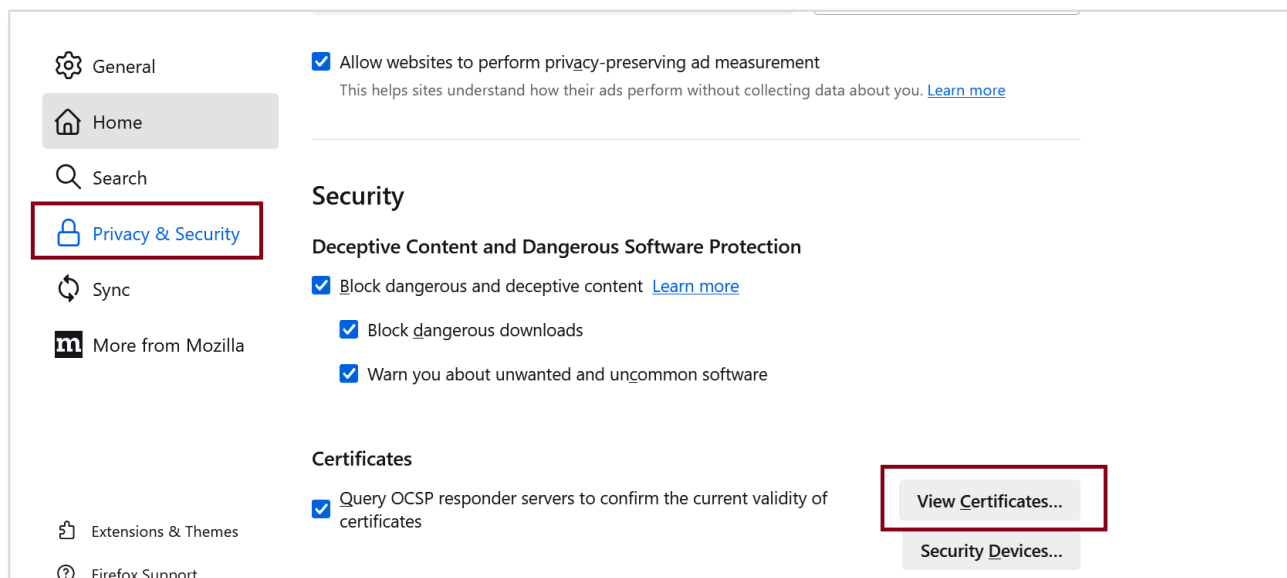
COMPANYA

4253c95af2dbc4647ec729ff16fc3eb5de...



4.2.3 Firefox

- Open Firefox → click on  (visible at the right top corner) → click on “Settings”
- Click on “Privacy & Security” → click on “View certificates”



- In the Certificate Manager window, search for the “Common Name (CN)” that you provided within your CSR. If the import was successful, you will be able to find the certificate with the CN as shown below under **“Your Certificates”**:

✓ EPEX SPOT			
COMPANYA	OS Client Cert Token	00:ED:8B:F...	Friday, 30 ...

5. MATS API App Users

5.1 Using client certificate in an API app

Once you generated the Keystore depending on which type of keystore file you use and which programming language, you must:

- pass client certificate keystore & password when opening an HTTPS url connection while logging into the system

Please note that the way the certificate and password must be technically passed can vary depending on the programming language you use.

5.1.1 Sample code

As a reference, please consult the 2FA sample codes provided in **Java** or **Python** as part of the MATS API package.

5.2 Using JKS or PKCS12 with SoapUI & Postman

With the Soap UI application, one can use any key store file format (.p12, .pfx, .jks) for a secure communication channel over TLS. Soap UI & Postman support all these file formats and work in the same way with all of them.

6. Annex - Technical Guidance

6.1 Examples of CSR and KeyStore files generation

6.1.1 Example of CSR generation using OpenSSL

Pre-requisite: OpenSSL must be installed to be able to generate a CSR.

- Command to generate a CSR and Private Key associated **with password protection for private key**:
openssl req -new -keyout [PrivateKeyPath] -out [CSRPath]
- Command to generate a CSR and Private Key associated **without password protection for private key**:
openssl req -new -nodes -keyout [PrivateKeyPath] -out [CSRPath]

Generation example on a Linux server (same command line when OpenSSL is installed on Windows):

a) With Password protection for private key

```
root@pluton:~# openssl req -new -keyout /tmp/MyPrivateKey.key -out /tmp/MyCsr.key
Generating a 2048 bit RSA private key
.....+++
writing new private key to '/tmp/MyPrivateKey.key'
Enter PEM pass phrase:
Verifying - Enter PEM pass phrase:
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [FR]:FR
State or Province Name (full name) [PARIS]:PARIS
Locality Name (eg, city) [PARIS]:PARIS
Organization Name (eg, company) [POWERNEXTSA]:POWERNEXTSA
Organizational Unit Name (eg, section) [DSI]:DSI
Common Name (e.g. server FQDN or YOUR name) []:POWERNEXTSA_Client001
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

b) Without Password:

```
root@pluton:~# openssl req -new -nodes -keyout /tmp/MyPrivateKey.key -out /tmp/MyCsr.k
Generating a 2048 bit RSA private key
.....+++
writing new private key to '/tmp/MyPrivateKey.key'
-----
You are about to be asked to enter information that will be incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [FR]:FR
State or Province Name (full name) [PARIS]:PARIS
Locality Name (eg, city) [PARIS]:PARIS
Organization Name (eg, company) [POWERNEXTSA]:POWERNEXTSA
Organizational Unit Name (eg, section) [DSI]:DSI
Common Name (e.g. server FQDN or YOUR name) []:POWERNEXTSA_Client002
Email Address []:

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

6.1.2 Example of PKCS12 keystore generation using OpenSSL

Pre-Requisites:

- OpenSSL must be installed
- you need to have the private Key file and the signed certificate (.pem file)

Command to generate a PKCS12 with private key and certificate:

```
openssl pkcs12 -in [CertificatePath] -inkey [PrivateKeyPath] -export -out [Pkcs12Path]
```

Example: Generation on a *Linux* server (same command line when OpenSSL is installed on Windows):

```
root@scolca001:~# openssl pkcs12 -in /tmp/ApiExemple.pem -inkey /tmp/ApiExemple.key -export -out /tmp/ApiExemple.p12
Enter pass phrase for /tmp/ApiExemple.key:
Enter Export Password:
Verifying - Enter Export Password:
root@scolca001:~# ls /tmp | grep p12
ApiExemple.p12
root@scolca001:~# █
```

6.1.3 Example of JKS keystore generation using Java keytool.exe

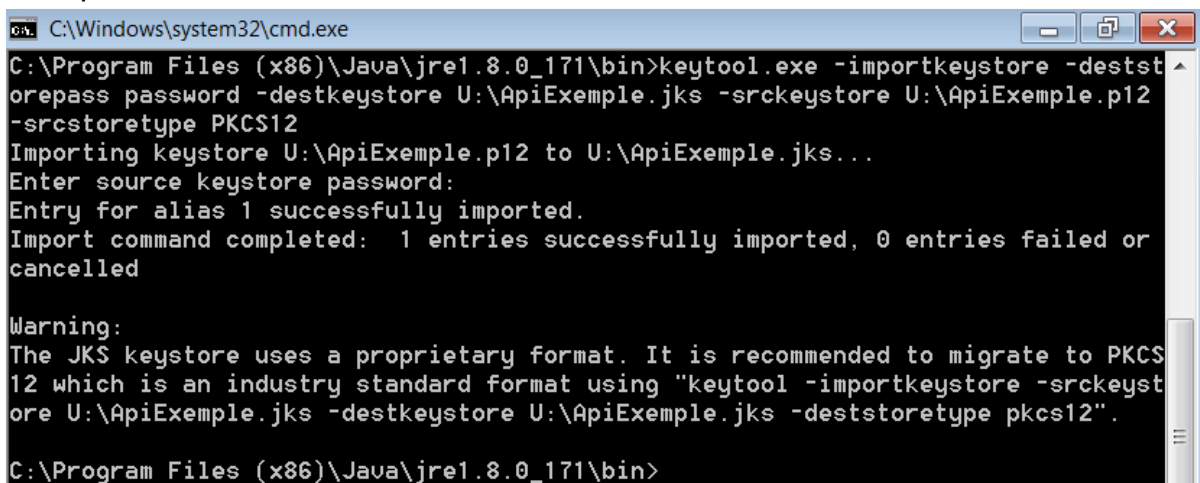
Pre-Requisites:

- Java must be installed
- you need to have a PKCS12 file.

Command to generate a JKS from PKCS12:

```
keytool.exe -importkeystore -deststorepass [WantedKeystorePassword] -destkeystore [DestinationKeystoreName.jks] -srckeystore [SourceKeyStore -srcstoretype PKCS12]
```

Example:



```
C:\Windows\system32\cmd.exe
C:\Program Files (x86)\Java\jre1.8.0_171\bin>keytool.exe -importkeystore -deststorepass password -destkeystore U:\ApiExemple.jks -srckeystore U:\ApiExemple.p12 -srcstoretype PKCS12
Importing keystore U:\ApiExemple.p12 to U:\ApiExemple.jks...
Enter source keystore password:
Entry for alias 1 successfully imported.
Import command completed: 1 entries successfully imported, 0 entries failed or cancelled

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 U:\ApiExemple.jks -destkeystore U:\ApiExemple.jks -deststoretype pkcs12".

C:\Program Files (x86)\Java\jre1.8.0_171\bin>
```

6.1.4 Example of PKCS12 keystore generation using OpenSSL for Mac Users

Pre-Requisites:

- OpenSSL must be installed
- The version of OpenSSL that is used for internal keystore of Mac OS **must be same as the** OpenSSL used for generating the PFX file
- If there is a conflict in OpenSSL version, then the command line with the keyword -legacy should be used as indicated below.
- you need to have the private Key file and the signed certificate (.pem file)

Command to generate a PKCS12 with private key and certificate for Mac OS users:

```
openssl pkcs12 -export -legacy -out [Pkcs12Path] -inkey [PrivateKeyPath] -in [CertificatePath]
```

6.2 Troubleshooting

6.2.1 How to debug a handshake issue

You have your signed certificate (.pem file) and private key, but do not success in connecting to our server because of an handshake issue: you can use the below approach.

- Command to check your TLS connection to our server. Simulation example:
 - *openssl s_client -showcerts -connect mats_simu_host_url:443 -tls1_2 -state -key <privateKey> -cert <PEMFile>*
 - In case of failure, please contact Market Operations

Note: In case you are using PYTHON and unable to establish the TLS connection:

- 1) Force the use of the TLS version 1.2.
- 2) Do not use a pkcs12 keystore –just pass the .pem file and your private key .key file as separate arguments.