EPEX SPOT
EUROPEAN POWER EXCHANGE

ETS API Certificates

Date: 18/05/2018

Place : Paris

Document release: v4

Content

1. Introduction

a. Audience

This document is intended for members who will use ETS API, EPEX SPOT Market Operators and EPEX SPOT IT Team.

b. Purpose

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

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

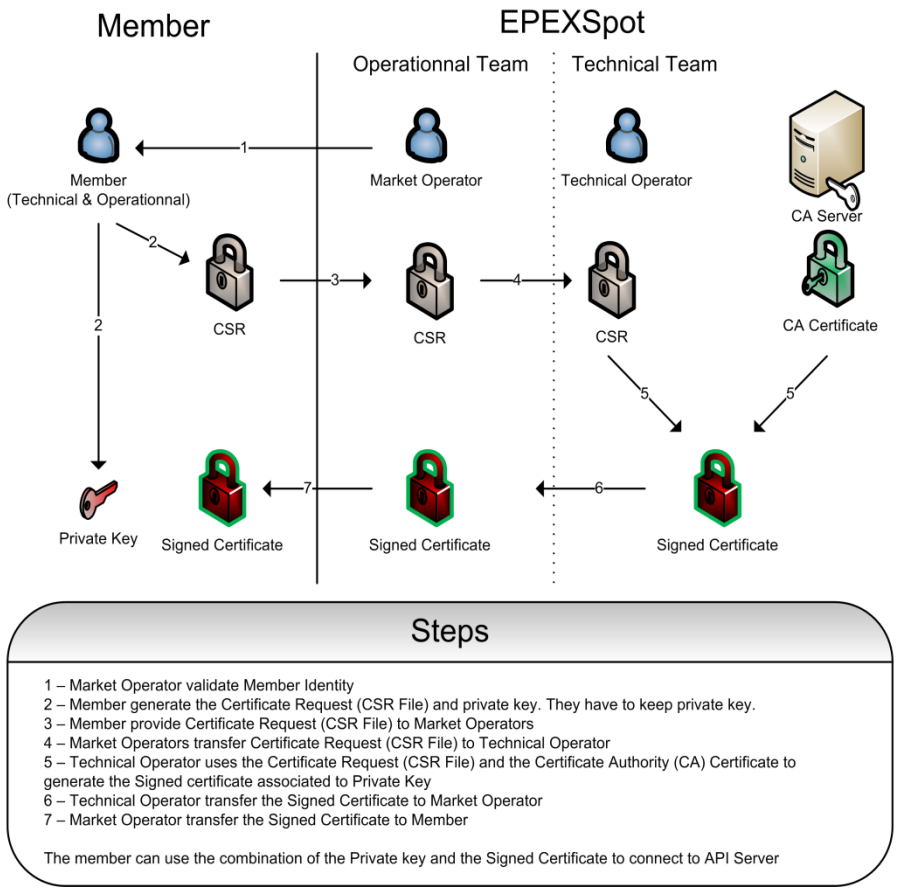
2. Certification rules and process

a. General information

A certificate includes a combination of a private key and a public key. The private part of the certificate should never be provided to anyone. If for any reason the private part of the certificate is shared, EPEX SPOT will not be able to guarantee the identity of the member.

b. Certification process

The following schema explains the certification process:



The exchanged files (CSR File and Signed Certificate file) are public and can be exchanged by email.

The only valid certificates will be signed by a Certificate Authority owned by EPEX SPOT.

c. Technical information

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

- All communications between ETS API Clients (member application) and ETS API Server are encrypted, using HTTPS
- Bi-lateral authentication system which require a Client certificate to connect

In order to guarantee compatibility, following solution is supported by ETS API:

- Protocol TLS 1.2
- SHA2 cryptographic hash function with RSA encryption for public key

Please note that other signing algorithms are not supported in this version.

d. Certificate Management Rules

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

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

Following conditions should be met for the creation of the CSR:

- Only ASCII characters are accepted
- Country Name (2 letter code) : Must respect the country of the company (Validated by Operators)
- Organization Name (eg, company): Must meet the company short name. The company short name is the one set by EPEX SPOT at member registration; please contact Market Operation if you do not remember your company short name
- Common Name (e.g. server FQDN or YOUR name): Client name must start with OrganizationName _ (OrganizationName is the same as previous field). If you have several certificates, this name must be unique

e. Revocation of certificate

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

- The member contacts the Market Operators, who validate member identity
- The member provides **Country Name / Organization Name / Common Name** combination which identifies the certificate to be revoked
- The member is contacted (by email) by the Market Operators to confirm the revocation of his certificate

Once revoked, a certificate cannot be used anymore.

f. Renewal of certificate

The certificates are valid for 1 Year. It is the member's responsibility to follow-up on it's expiration.

The validity period of a certificate cannot be extended and a new CSR should be provided to EPEXSPOT to generate a new signed certificate. The same **"Country Name / Organization Name / Common Name"** combination can be used for the new CSR.

g. Example of CSR generation using OpenSSL

Requirement: OpenSSL has to 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:

With Password

```
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 []:
```

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 []:
```

h. Example of PKCS12 certificate container using OpenSSL

PKCS12: A PKCS12 format is a certificate container which can contain Certificate File and the private Key.

Requirement: OpenSSL has to be installed; you need to have the private Key file and the Certificate

Command to generate a PKCS12 with private key and certificate:

openssl pkcs12 -in [CertitifactePath] -inkey [PrivateKeyPath] -export -out [Pkcs12Path]

Generation example on a Linux server:

```
root@scolca001:~# openssl pkcs12 -in /tmp/ApiExemple.pem -inkey /tmp/ApiExe
mple.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:~#
```

i. Example of JKS certificate container using Java keytool.exe

JKS: A JKS format is a certificate container which can contain Certificate File and the private Key. Contrary to PKCS12 which is explain before, this format is Java proprietary format.

Requirement: Java has to 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: