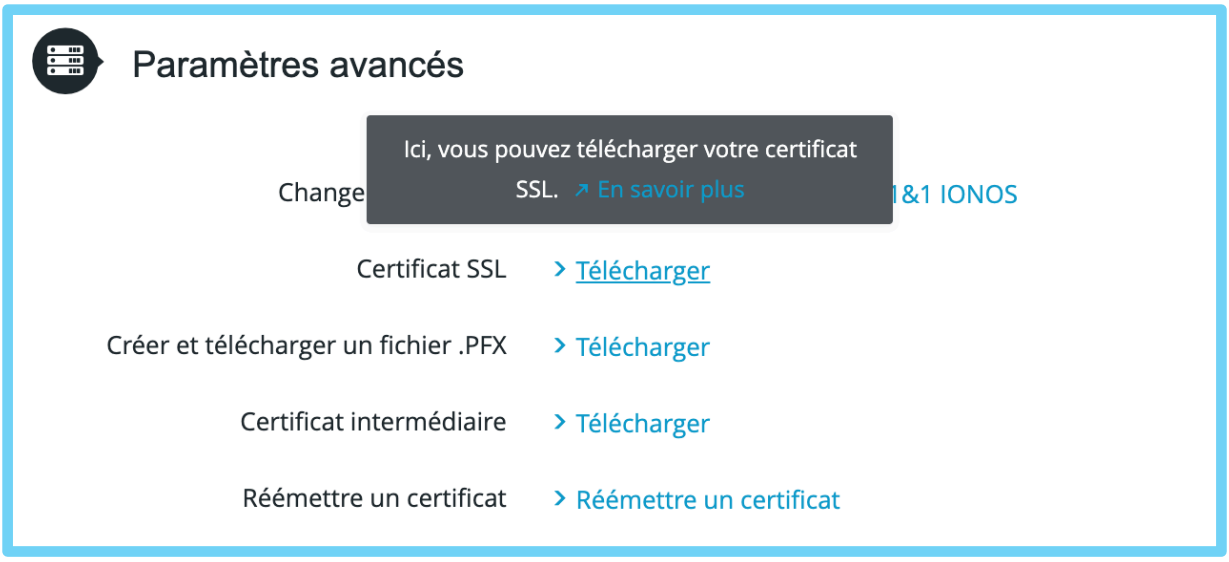
- **Step (0)-** : Download SSL certificate



➔ You end up downloading a .cer file (**your_ssl_certificate.cer**) on your favourite domain provider
➔ You already have downloaded and stored securely the private key (**yourdomain_ssl.key**) when issuing the certificate (for security reason, not stored on the IONOS server). If not, re-emit…

- **Step (1)-** : convert the .cer file into .pem format via OpenSSL using the following command line

```
openssl x509 -in your_ssl_certificate.cer -out your_ssl_certificate.pem
```

legacy:

```
openssl x509 -in aps.cer -inform DER -out aps.pem -outform PEM
```

- **Step (2)-** : use the .pem file and your private .key to generate .p12 (**yourdomain_ssl.p12**) file with OpenSSL using the following command line:

```
openssl pkcs12 -export -out yourdomain_ssl.p12 -inkey yourdomain_ssl.key -in your_ssl_certificate.pem
```

- **Step (3)-** : push the key in the keystore using JAVA keytool with the following command line (**Using JDK 1.6 or later**):

Use the same password than in step 2 (see warning below)

```
keytool -importkeystore -srckeystore yourdomain_ssl.p12 -srcstoretype pkcs12 -destkeystore clientcert.jks -deststoretype JKS
```

As specified by the API, the KeyManagerFactory.init method takes in the password used to retrieve the keys from the keystore. Since there is only one password parameter, it is expecting that the password for all the keys are identical. If a different password is used for one of the keys, then you get the error you saw as the password is incorrect for that particular keystore entry.

The simplest solution for you would be to use the same password for all the entries in the keystore. If you are set on maintaining different passwords for each entry, then you may have to look into building your own custom security elements, e.g., KeyManager.

(source: https://stackoverflow.com/questions/4926290/java-keystore-and-password-settings)