Keyfactor EJBCA Enterprise

nShield® HSM Integration Guide

2026-03-16

Table of Contents

Chapter 1. Introduction

Keyfactor EJBCA Enterprise is a certificate management solution that is ready for post-quantum cryptography (PQC). It deploys fast, runs anywhere, and scales to meet any architecture. Entrust nShield HSMs provide FIPS or Common Criteria certified solutions to securely generate, encrypt, and decrypt cryptographic keys.

The nShield HSM provides enhanced security to Keyfactor EJBCA Enterprise in the management of cryptographic keys and certificates. In addition, the nShield HSM supports various PQC algorithms. This guide describes how to integrate Keyfactor EJBCA Enterprise with an nShield HSM.

1.1. Product configurations

Entrust has successfully tested nShield HSM integration with Keyfactor EJBCA Enterprise in the following configurations:

Product	Version
Keyfactor EJBCA Enterprise Cloud - 8x5 Support	v9.4.2

1.2. Supported nShield features

Entrust has successfully tested nShield HSM integration with the following features:

Feature	Support
Softcards	Yes
Module-only key	Yes
OCS cards	Yes
nSaaS	Supported but not tested

1.3. Supported nShield hardware and software versions

Entrust has successfully tested with the following nShield HSM hardware and software versions:

1.3.1. nShield 5c

Security World Software	Firmware	Netimage	OCS	Softcard	Module
13.9.3 (STS 4)	13.8.4	13.9.3	✓	✓	✓

1.3.2. Connect XC

Security World Software	Firmware	Netimage	OCS	Softcard	Module
13.9.3 (STS 4)	13.8.3	13.9.3	✓	✓	✓

1.4. Requirements

- Access to the Entrust TrustedCare Portal. This portal is available to customers under maintenance. To request an account, contact nshield.support@entrust.com.
- An Entrust nShield HSM.

Familiarize yourself with the Keyfactor EJBCA Cloud AWS documentation and the nShield Security World documentation.

You must also understand:

- The importance of a correct quorum for the Administrator Card Set (ACS).
- Whether Operator Card Set (OCS) protection or Softcard protection is required.
  - If OCS protection is to be used, a 1-of-N quorum must be used.
- Whether your Security World must comply with FIPS 140 Level 3 or Common Criteria standards.

  If using FIPS 140 Level 3, it is advisable to create an OCS for FIPS authorization. The OCS can also provide key protection for the Vault master key. For more information see FIPS 140 Level 3 compliance.
- Whether to instantiate the Security World as recoverable or not.

Chapter 2. System deployment

For this integration test, Keyfactor EJBCA Enterprise Cloud - 8x5 Support was deployed as an EC2 instance in AWS. The HSMs were located in an Entrust facility. The Amazon Virtual Private Cloud security group and the Entrust firewall were configured as needed. Operator Card Set (OCS) authorization was presented remotely using the Entrust Trusted Verification Device (TVD).

Chapter 3. Install and configure the Entrust nShield HSM

3.1. Install the nShield Security World Software

1. Sign in to the Keyfactor EJBCA Enterprise CLI.
2. Install the Security World software. For detailed instructions, see the nShield Security World Software v13.9.3 Installation Guide.
3. Add the Security World utilities path to the system path. This path is typically `/opt/nfast/bin`. Do so by creating the `/etc/profile.d/nfast.sh` file:

```
$ cat /etc/profile.d/nfast.sh
# Entrust nShield HSM
export PATH=$PATH:/opt/nfast/bin
```

4. Open firewall port 9004 for the Entrust nShield HSM connections. Alternatively, in AWS, add a TCP inbound/outbound rule connecting to the HSM to the EC2 security group.

```
$ sudo firewall-cmd --permanent --add-port=9004/tcp
$ sudo firewall-cmd --reload
```

5. If using Remote Administration, open firewall port 9005 for the Entrust nShield Trusted Verification Device (TVD). Alternatively, in AWS, add inbound and outbound TCP rules between the EC2 security group and the machine hosting the TVD.
6. Run the `enquiry` utility to confirm the Security World is **operational**.

```
$ /opt/nfast/bin/enquiry
Server:
 enquiry reply flags  none
 enquiry reply level  Six
 serial number
 mode                 operational
```

```
version             13.6.14
...
```

3.2. Install the Entrust nShield HSM

Install the nShield Connect HSM locally, remotely, or remotely via the serial console. Condensed instructions are available in Entrust TrustedCare Portal.

- How To: Locally Set up a new or replacement nShield Connect.
- How To: Set up new or replacement nShield 5s.
- How To: Remotely Setup a new or replacement nShield Connect.
- How To: Remotely Setup a new or replacement nShield Connect XC Serial Console Model.

For detailed instructions see the nShield v13.9.3 Hardware Install and Setup Guides.

3.3. Enroll the Entrust nShield HSM

1. Inform the HSM of the client's location.

   In this integration the client is the Keyfactor EJBCA Enterprise AWS EC2 instance. For instructions see Configuring the nShield HSM to use the client.

   If it is a high-availability setup, repeat the client configuration for each HSM.

2. Enroll the Keyfactor EJBCA Enterprise AWS EC2 instance as a client of the HSM.

   For instructions see Configuring client computers to use the nShield HSM.

   If it's a high-availability setup, repeat the enrollment for each HSM.

3. Run the `enquiry` utility to confirm the HSM is **operational**:

```
$ enquiry
Server:
 enquiry reply flags  none
 enquiry reply level  Six
 serial number        8FE1-B519-C5AA 6308-03E0-D947
 mode                 operational
 version              13.6.14
 ...
Module #1:
 enquiry reply flags  UnprivOnly
 enquiry reply level  Six
 serial number        8FE1-B519-C5AA
 mode                 operational
 version              13.4.5
 ...
 module type          nShield 5c
```

```
 ...
Module #2:
 enquiry reply flags  UnprivOnly
 enquiry reply level  Six
 serial number        6308-03E0-D947
 mode                 operational
 version              12.72.4
 ...
 module type          nShield Connect XC
 ...
```

3.4. Create a security world

1. Create your Security World, if one does not already exist, or copy an existing one.

   Follow your organization's security policy for this. For more information see Create a new Security World.

   ACS cards cannot be duplicated after the Security World is created. You may want to create extras in case of a card failure or a lost card.

2. Confirm the Security World is **Usable** with `nfkminfo`:

```
$ /opt/nfast/bin/nfkminfo
World
 generation  2
 state       0x37270008 Initialised Usable ...
 ...
Module #1
 generation 2
 state      0x2 Usable
 ...
Module #2
 generation 2
 state      0x2 Usable
 ...
```

3.5. Select the protection method

OCS, Softcard, or Module protection can be used to authorize access to the keys protected by the HSM. Typically, an organization's security policies dictate the use of one of these methods.

- Operator Cards Set (OCS) are smartcards that are presented to the physical smartcard reader of an HSM. For more information on OCS use, properties, and k-of-N values, see Operator Card Sets (OCS).
- Softcards are logical tokens (passphrases) that protect the key and authorize its use. For more information on Softcard use, see Softcards.

- Module protection has no passphrase.

Follow your organization's security policy to select an authorization access method.

1. Create the `/opt/nfast/cknfastrc` file containing the nShield PKCS #11 library environment variables per the selection above.

   For example:

   ```
   # Enable Softcard protection
   CKNFAST_LOADSHARING=1

   # Enable Module protection
   CKNFAST_FAKE_ACCELERATOR_LOGIN=1

   # OCS Preload file location and card set state
   NFAST_NFKM_TOKENSFILE=/opt/nfast/preloadtoken
   CKNFAST_NONREMOVABLE=1

   # PKCS #11 log level and file location
   CKNFAST_DEBUG=3
   CKNFAST_DEBUGFILE=/opt/nfast/log/pkcs11.log
   ```

2. Change ownership of the `/opt/nfast/cknfastrc` file to `nfast`:

   ```
   $ ls -al /opt/nfast/cknfastrc
   -rw-rw-rw-. 1 root root 324 Apr  3 16:12 /opt/nfast/cknfastrc

   $ sudo chown nfast:nfast /opt/nfast/cknfastrc

   $ ls -al /opt/nfast/cknfastrc
   -rw-rw-rw-. 1 nfast nfast 324 Apr  3 16:12 /opt/nfast/cknfastrc
   ```

3. Make the `/opt/nfast/log/pkcs11.log` file writable:

   ```
   $ sudo touch /opt/nfast/log/pkcs11.log

   $ sudo chmod 664 /opt/nfast/log/pkcs11.log
   ```

4. Restart the nShield service:

   ```
   $ sudo /opt/nfast/sbin/init.d-ncipher restart
   ...
   ```

3.6. Create the OCS

1. Edit the `/opt/nfast/kmdata/config/cardlist` file by adding the serial number of the cards to be presented or the wildcard "*".
2. Run the `createocs` utility as described below, entering a passphrase at the prompt.

Follow your organization's security policy for the values K/N. Use the same passphrase for all the OCS cards in the set (one for each person with access privilege, plus the spares). In this example note that `slot 2`, remote via a TVD, is used to present the card.

After an OCS card set has been created, the cards cannot be duplicated. You may want to create extras in case of a card failure or a lost card.

The preload utility loads OCS onto the HSM. This feature makes the OCS available for use after being physically removed from the HSM for safe storage or other reasons. Add the **-p** (persistent) option to the command below to have authentication after the OCS card has been removed from the HSM front panel slot or from the TVD.

```
$ /opt/nfast/bin/createocs -m1 -s2 -N testOCS -Q 1/1

FIPS 140-2 level 3 auth obtained.

Creating Cardset:
 Module 1: 0 cards of 1 written
 Module 1 slot 0: Admin Card #1
 Module 1 slot 2: empty
 Module 1 slot 3: empty
 Module 1 slot 2: blank cardSteps:

 Module 1 slot 2:- passphrase specified - writing card
Card writing complete.

cardset created; hkltu = a165a26f929841fe9ff2acdf4bb6141c1f1a2eed
```

In the example above, the authentication provided by the OCS is non-persistent and only available while the OCS card is inserted in the HSM front panel slot or the TVD.

3. Verify the OCS using `nfkminfo` or `rocs`:

nfkminfo

```
$ /opt/nfast/bin/nfkminfo -c
Cardset list - 2 cardsets:  (P)ersistent/(N)ot, (R)emoteable/(L)ocal-only
 Operator logical token hash               k/n timeout  name
 edb3d45a28e5a6b22b033684ce589d9e198272c2  1/5  none-NL testOCS
```

rocs

```
$ /opt/nfast/bin/rocs
`rocs' key recovery tool
Useful commands: `help', `help intro', `quit'.
rocs> list cardset
No. Name                    Keys (recov) Sharing
  1 testOCS                 0 (0)        1 of 5
rocs> quit
```

3.7. Create the Softcard

1. Enable Softcard protection as described in Select the protection method.
2. Run the `ppmk` utility, and enter a passphrase at the prompt:

```
$ /opt/nfast/bin/ppmk -n testSC
Enter new pass phrase:
Enter new pass phrase again:
New softcard created: HKLTU 644529aad18eeed9de372779e829f48757e617cd
```

3. Verify the Softcard using `nfkminfo` or `rocs`:

nfkminfo

```
$ /opt/nfast/bin/nfkminfo -s
SoftCard summary - 1 softcards:
 Operator logical token hash               name
 644529aad18eeed9de372779e829f48757e617cd  testSC
```

rocs

```
$ /opt/nfast/bin/rocs
`rocs' key recovery tool
Useful commands: `help', `help intro', `quit'.
rocs> list cardsets
No. Name                    Keys (recov) Sharing
  1 testOCS                 0 (0)        1 of 5
  2 testSC                  0 (0)        (softcard)
rocs> quit
```

Chapter 4. Configure Keyfactor EJBCA Enterprise to use the Entrust nShield HSM

The configuration is done using the Keyfactor EJBCA Enterprise CLI.

- Add the Keyfactor EJBCA Enterprise users to the nfast group
- Verify that the PKCS #11 library is available
- Configure Keyfactor EJBCA Enterprise to recognize the nShield PKCS#11 library

4.1. Add the Keyfactor EJBCA Enterprise users to the nfast group

1. Sign in to the Keyfactor EJBCA Enterprise CLI.
2. Add the Keyfactor EJBCA Enterprise user to the nfast group.

```
$ sudo gpasswd -a ec2-user nfast
Adding user ec2-user to group nfast
```

3. Add the `wildfly` user to the nfast group.

```
$ sudo gpasswd -a wildfly nfast
Adding user wildfly to group nfast
```

4.2. Verify that the PKCS #11 library is available

1. If using OCS protection, present the OCS to the HSM.
2. Sign in to the Keyfactor EJBCA Enterprise CLI.
3. Run the `ckcheckinst` utility to test the library.
    - Enter the slot number corresponding to the protection method used.
    - Enter the OCS or Softcard passphrase when prompted.

```
$ ckcheckinst
PKCS#11 library interface version 2.40
                           flags 0
                  manufacturerID "nCipher Corp. Ltd               "
              libraryDescription "nCipher PKCS#11 13.9.3-334-66c0b"
          implementation version 13.09
        Loadsharing and Failover enabled

Slot  Status            Label
====  ======            =====
   0  Fixed token       "loadshared accelerator          "
```

```
   1  Fixed token       "testOCS                        "
   2  Soft token        "testSC                         "


No removable tokens present.
Please insert an operator card into at least one available slot and enter 'R' retry.
If you have not created an operator card or there are no physical slots,
 enter a fixed token slot number,
 or 'E' to exit this program and create a card set before continuing.

Enter a fixed token slot number, 'R'etry or 'E'xit: 1
Using slot number 1.

Please enter the passphrase for this token (No echo set).
Passphrase:

Test                      Pass/Failed
----                      -----------

1 Generate RSA key pair   Pass
2 Generate DSA key pair   Pass
3 Encryption/Decryption   Pass
4 Signing/Verification    Pass

Deleting test keys         ok

PKCS#11 library test successful.
```

4.3. Configure Keyfactor EJBCA Enterprise to recognize the nShield PKCS#11 library

1. Sign in to the Keyfactor EJBCA Enterprise CLI.
2. Create a backup of the `web.properties` configuration file:

```
$ sudo cp /opt/ejbca_ee_9_4_2/conf/web.properties /opt/ejbca_ee_9_4_2/conf/web.properties.bak
```

3. Edit the `web.properties` configuration file by adding the path to the nShield PKCS #11 library.

   See the `/opt/ejbca_ee_9_4_2/conf/samples/web.properties.sample` sample file for the path to the nShield PKCS #11 library.

```
$ sudo cat /opt/ejbca_ee_9_4_2/conf/web.properties
...
# Entrust nShield HSM
cryptotoken.p11.lib.40.name=nCipher
cryptotoken.p11.lib.40.file=/opt/nfast/toolkits/pkcs11/libcknfast.so
...
```

4. Comment out the paths to crypto libraries that are not installed.
5. Restart the EJBCA service:

```
$ sudo service wildfly restart
Redirecting to /bin/systemctl restart wildfly.service
```

Chapter 5. Test the integration

Testing is done using both the Keyfactor EJBCA Enterprise CLI and the Web GUI.

- Create the Crypto Token in the EJBCA GUI
- Generate the keys in the EJBCA GUI
- Verify the generated keys using the EJBCA CLI

5.1. Create the Crypto Token in the EJBCA GUI

1. If using OCS protection, present the OCS to the HSM.
2. Browse to the Keyfactor EJBCA Enterprise GUI.
3. Select **EJBCA Administration Web**.
4. In the toolbar, select **CA Functions** > **Crypto Tokens**.
5. Select **Add**.
6. Enter the information as shown and then select **Save**.

   The **Authentication Code** is the OCS or Softcard passphrase.

PKCS#11 NG is selected for **Type** because it is exclusive to EJBCA Enterprise and Keyfactor's commercial offerings like EJBCA Cloud. It is not available in the open-source EJBCA Community Edition, which uses the basic Java PKCS #11 provider. NG has lower-level control over the PKCS #11 API, for example, better session management, session restarting, and handling of non-standard behaviors. NG also has broader algorithm support including PQC ML-DSA and EdDSA, which allows this integration to work.

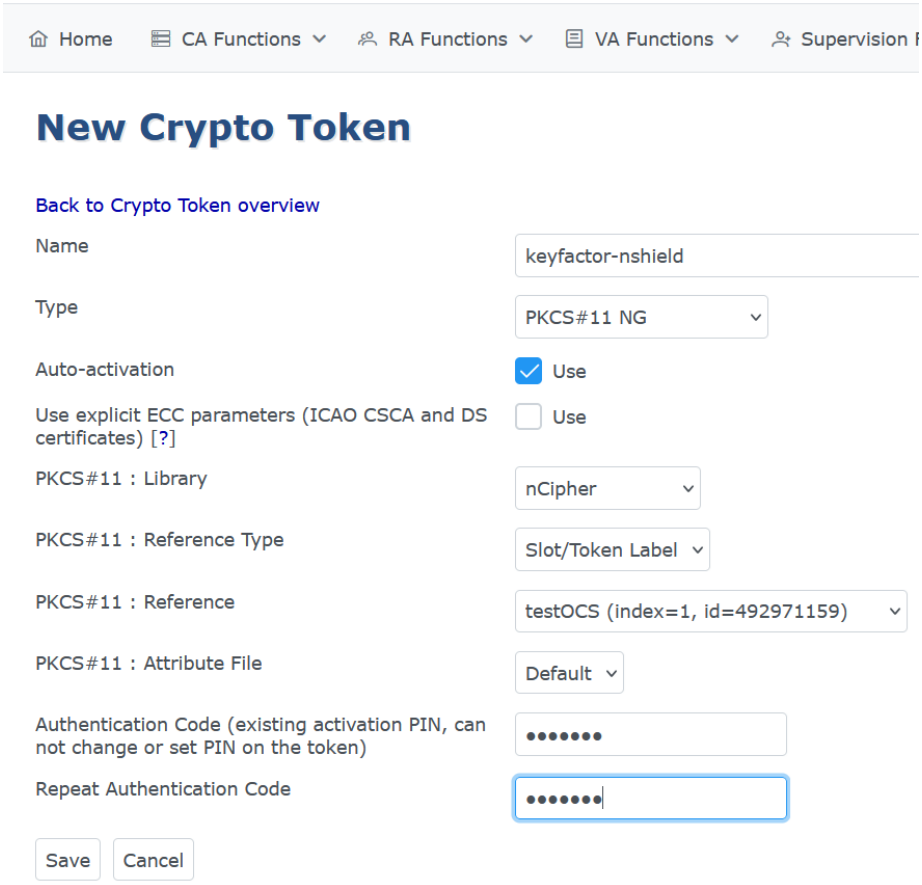


Figure 1. An example using OCS protection:

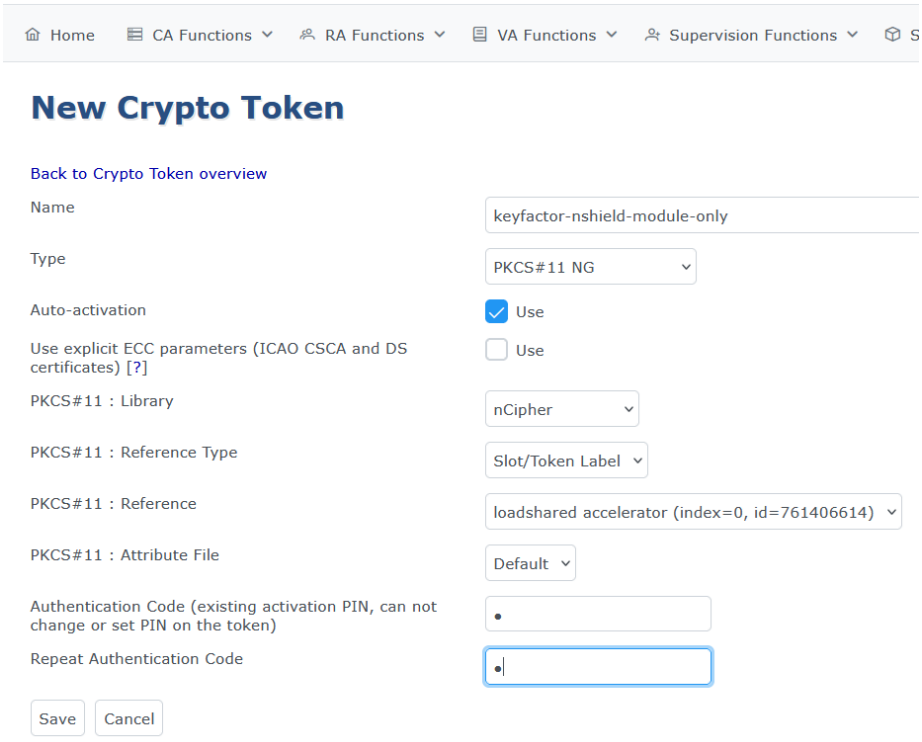
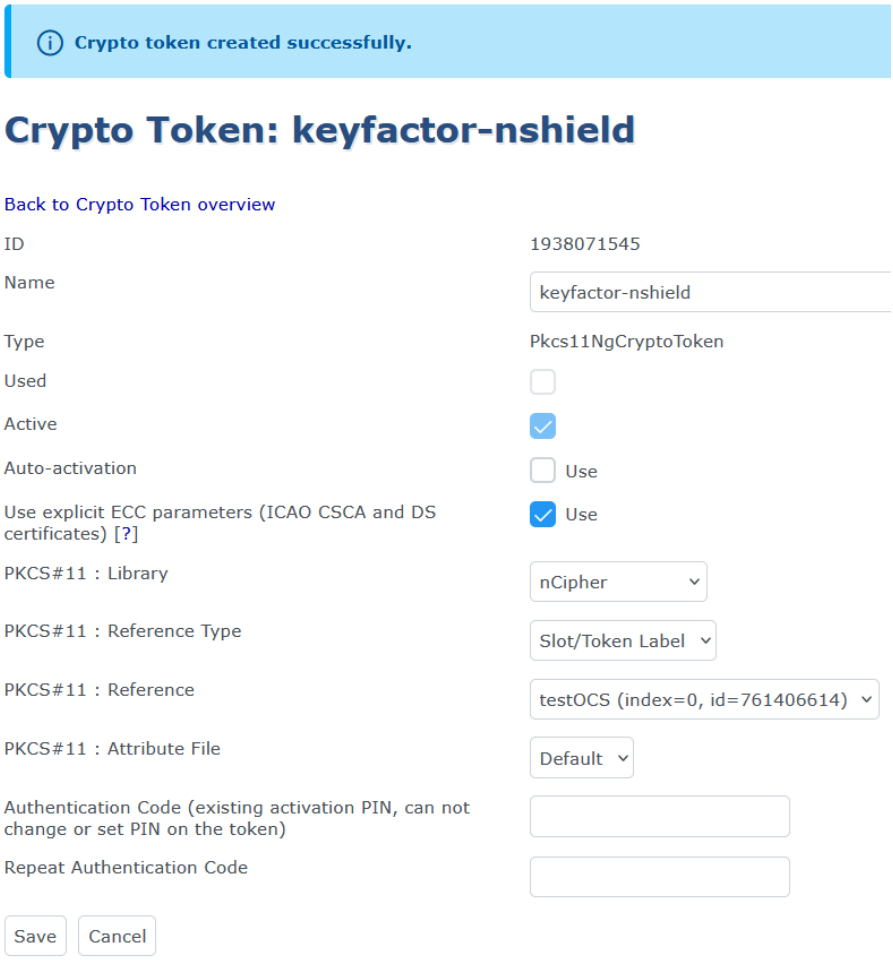


Figure 2. An example using module only protection:

For module only protection, enter any random string for **Authentication Code**.

7. You can now check the crypto token that was created.

   For example:



5.2. Generate the keys in the EJBCA GUI

1. Navigate to the Keyfactor EJBCA Enterprise GUI.
2. Select **EJBCA Administration Web**.
3. In the toolbar, select **CA Functions** > **Crypto Tokens**.
4. In the **List of Crypto Tokens**, locate the token to be used to create the keys.
5. Under **Actions**, select **Edit**.
6. Scroll down and select **Generate new key pair**.
7. Generate the following keys:

Name	Algorithm	Key Usage
signKey	ML-DSA-87	Sign / Verify
certSignKey	ML-DSA-87	Sign / Verify

For example:

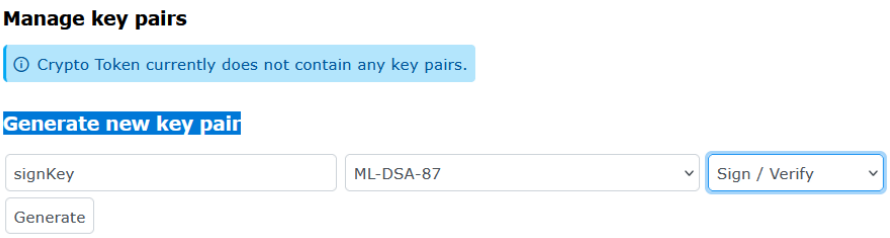


Figure 3. Generate new key pair

Figure 4. New keys generated

5.3. Verify the generated keys using the EJBCA CLI

1. Sign in to the Keyfactor EJBCA Enterprise CLI.
2. Verify the generated keys using the `nfkminfo` utility or the `rocs` utility:

nfkminfo

```
$ /opt/nfast/bin/nfkminfo -l

Keys protected by cardsets:
 key_pkcs11_uc321b143185fc939504893270ed821ba4ed38319a-879d31d0c3985f6ff552cd6b330f638fccda1ed2 'priv-
certSignKey'
 key_pkcs11_uc321b143185fc939504893270ed821ba4ed38319a-91bf2395ec1a3890a14a6efdb88e3ab568638fde 'priv-
signKey'
```

rocs

```
$ /opt/nfast/bin/rocs
`rocs' key recovery tool
Useful commands: `help', `help intro', `quit'.
rocs> list keys
  No. Name             App    Protected by
    1 priv-signKey     pkcs11 testOCS
    2 priv-certSignKey pkcs11 testOCS
```

```
rocs> quit
```

Chapter 6. Example of a root CA with PQC

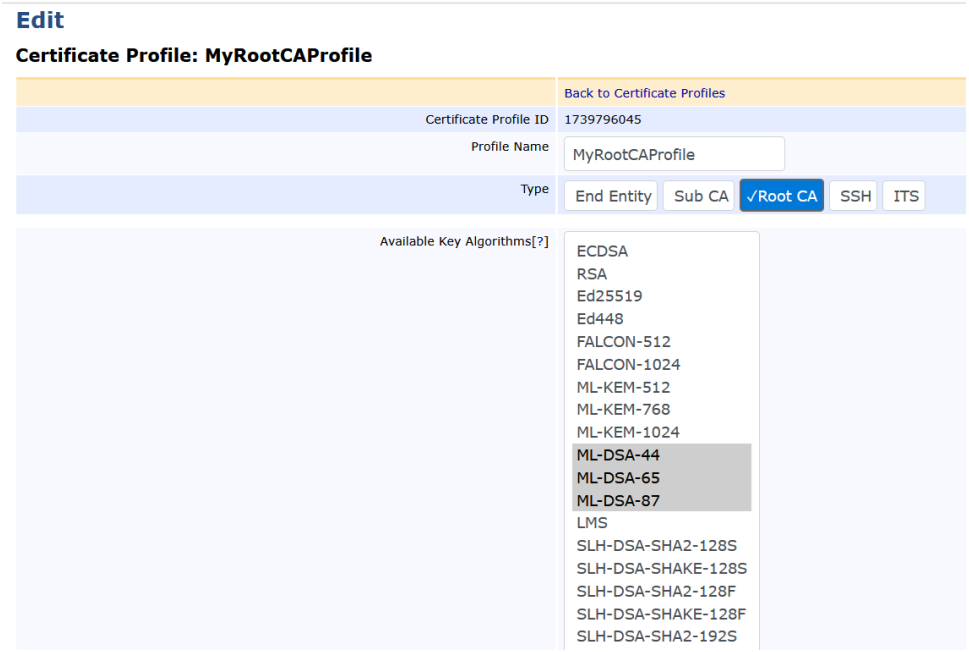
For reference, see Tutorial - Create your first Root CA using EJBCA.

- Create a certificate profile
- Create a Crypto Token
- Create the CA

6.1. Create a certificate profile

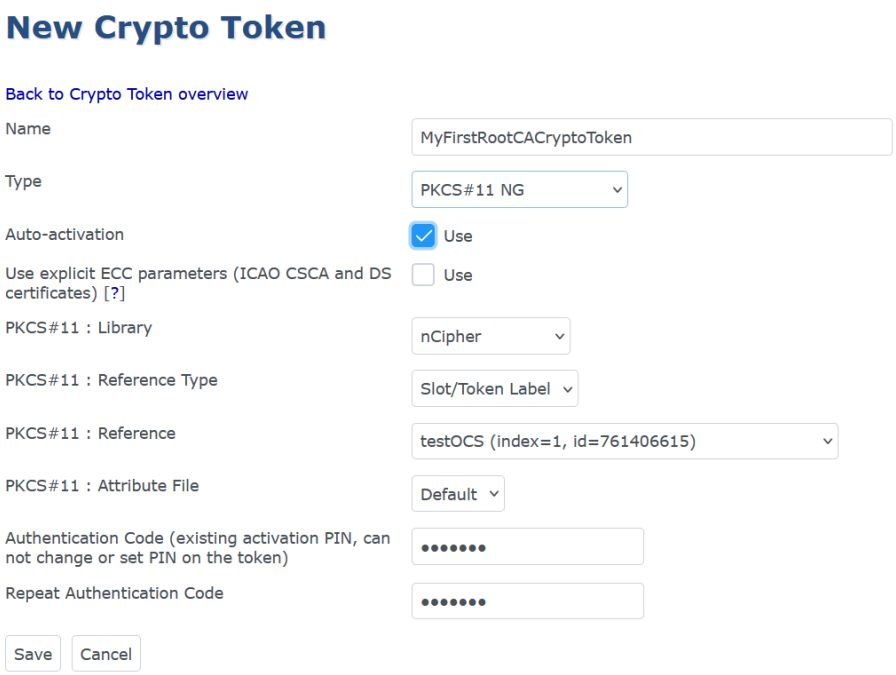
Create a certificate profile as described in the Keyfactor documentation: Create certificate profile.

For example:

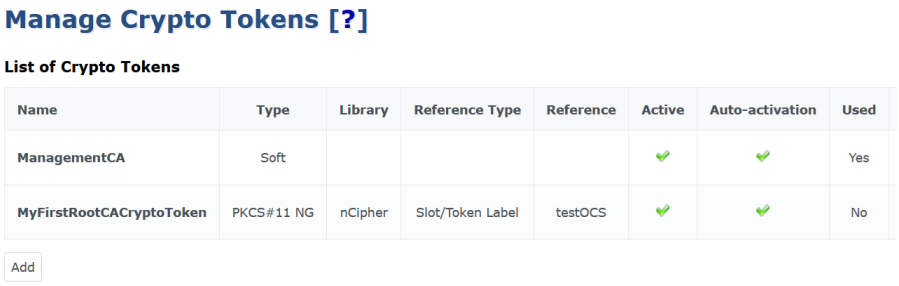


6.2. Create a Crypto Token

1. Create a crypto token as described in the Keyfactor documentation: Create crypto token.



The crypto token is added to the list.



2. Generate the following keys:

Name	Algorithm	Key Usage
myFirstRootCaSignKey0001	ML-DSA-87	Sign / Verify
myFirstRootCaAltSignKey0001	ML-DSA-44	Sign / Verify
myFirstRootCaEncryptKey0001	RSA 4096	Sign and Encrypt

Manage key pairs

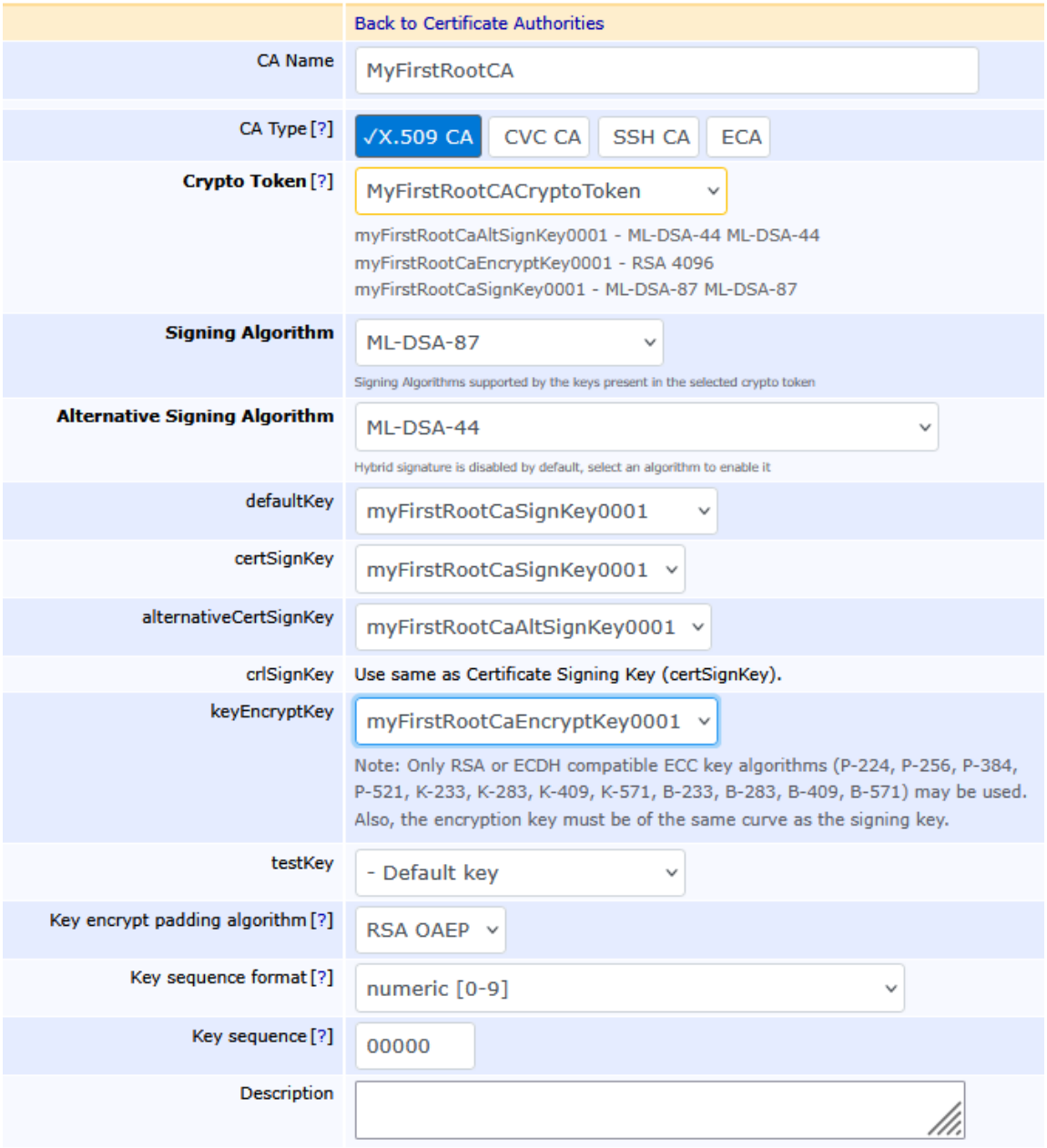
	Alias	Key Algorithm	Key Specification	SubjectKeyID	Key Usage	Action
☐	myFirstRootCaAltSignKey0001	ML-DSA-44	ML-DSA-44	6173745f6dba6f15cc14c949b77cccc404ebaf14	SIGN	Test Remove Download Public Key
☐	myFirstRootCaEncryptKey0001	RSA	4096	6724461cfd678f41d5d89e8da08a2638c6845ff2	SIGN_ENCRYPT	Test Remove Download Public Key
☐	myFirstRootCaSignKey0001	ML-DSA-87	ML-DSA-87	9851fa2db073982bf549d9d8ee903fef7e451566	SIGN	Test Remove Download Public Key

Remove selected

6.3. Create the CA

1. Create the root CA as described in the Keyfactor documentation: Create CA.

   For example:



2. The new CA appears in the list of certificate authorities:

3. Select the **View** icon to display the new CA details.

	Back to Certificate Authorities
CA Name	MyFirstRootCA
CA ID	-1716503095
CA Type [?]	X509
Crypto Token [?]	MyFirstRootCACryptoToken
Signing Algorithm	ML-DSA-87
Alternative Signing Algorithm	ML-DSA-44
defaultKey	myFirstRootCaSignKey0001
certSignKey	myFirstRootCaSignKey0001
alternativeCertSignKey	myFirstRootCaAltSignKey0001
crlSignKey	myFirstRootCaSignKey0001
keyEncryptKey	myFirstRootCaEncryptKey0001
testKey	myFirstRootCaSignKey0001
Key encrypt padding algorithm [?]	RSA OAEP
Key sequence format [?]	numeric [0-9]
Key sequence [?]	00000
Description	

Chapter 7. Example of generation of public certificate

In this example, a web server named MyWebServer will request a public certificate from the KeyFactor EJBCA Enterprise CA created in Example of a root CA with PQC.

- Create the certificate request
- Sign the certificate request

7.1. Create the certificate request

1. Create the private key for MyWebServer, for example, by logging in to MyWebServer and running the following command to create a key named `MyWebServer.key`:

```
>openssl genrsa -out MyWebServer.key 2048
```

2. Create the openssl configuration file, for example, save the following configuration as `MyWebServer.cnf`.

```
[ req ]
default_bits       = 2048
distinguished_name = req_distinguished_name
req_extensions     = req_ext
prompt             = no

[ req_distinguished_name ]
C  = US
ST = Florida
L  = Sunrise
O  = Example Company
OU = IT Department
CN = www.MyWebServer.com
emailAddress = info@MyWebServer.com

[ req_ext ]
subjectAltName = @alt_names

[ alt_names ]
DNS.1 = www.MyWebServer.com
DNS.2 = MyWebServer.com
```

3. Create the certificate request:

```
>openssl req -new -key MyWebServer.key -out MyWebServer.csr -config MyWebServer.cnf
```

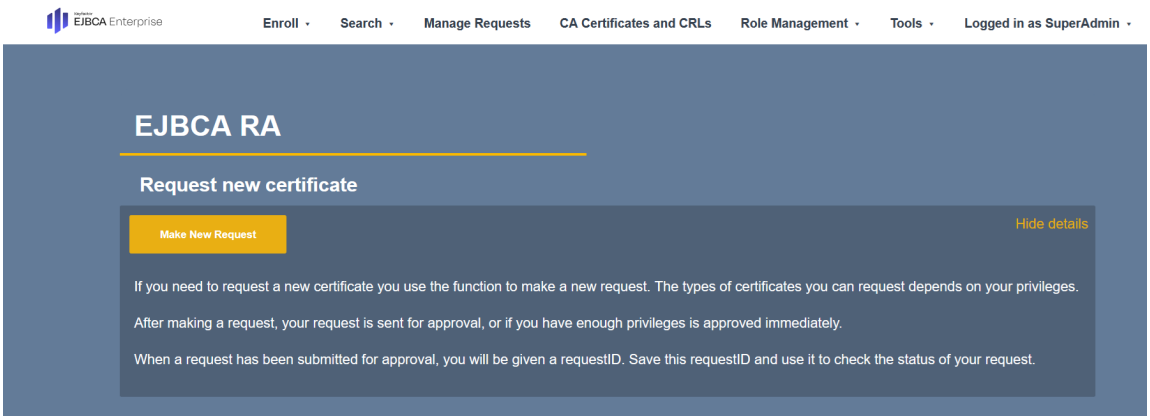
4. Display the content of the certificate request:

```
>openssl req -new -key MyWebServer.key -out MyWebServer.csr -config MyWebServer.cnf
```

```
C:\Users\ramired1\Downloads>openssl req -text -noout -verify -in MyWebServer.csr
Certificate request self-signature verify OK
Certificate Request:
    Data:
        Version: 1 (0x0)
        Subject: C = US, ST = Florida, L = Sunrise, O = Example Company, OU = IT Department, CN =
www.MyWebServer.com, emailAddress = info@MyWebServer.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:e3:27:b2:02:1c:01:28:81:45:37:0e:d3:c7:de:

                    ...

                    22:6a:09:80:39:b1:ac:dc:18:8c:50:32:59:d1:5e:
                    c3:8b
                Exponent: 65537 (0x10001)
        Attributes:
            Requested Extensions:
                X509v3 Subject Alternative Name:
                    DNS:www.MyWebServer.com, DNS:MyWebServer.com
    Signature Algorithm: sha256WithRSAEncryption
    Signature Value:
        0b:73:13:3b:ef:dd:35:bd:6e:68:43:27:79:34:b3:1d:f1:87:

        ...

        98:02:d0:b7:e2:df:22:b7:cb:d3:67:07:f0:a0:65:30:9e:c1:
        01:dd:22:a7
```

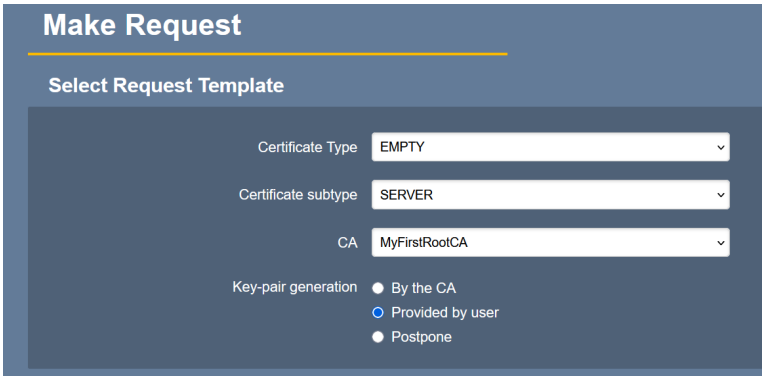
7.2. Sign the certificate request

1. In a browser, navigate to http://<your-server-ip>/ejbca/ra.



2. Select **Make New Request**.
3. Select the following options:
    - Certificate Type: **EMPTY**
    - Certificate subtype: **SERVER**
    - CA: **MyFirstRootCA**

- Key-pair generation: **Provided by the user**



4. Scroll down to **Upload CSR** and upload the CSR created in Create the certificate request.
5. Scroll down to **Provide User Credentials** and enter your credentials.
6. Scroll down and select **Download PEM**. A certificate file named `www.MyWebServer.com.pem` is downloaded to your computer.
7. Display the public certificate content:

```
>openssl x509 -text -noout -verify -in www.MyWebServer.com.pem
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            71:d2:24:9a:e0:e6:07:62:af:39:40:82:8b:6a:c2:4e:14:04:55:3e
        Signature Algorithm: 2.16.840.1.101.3.4.3.19
        Issuer: C = SE, O = Keyfactor Community, CN = MyFirstRootCA
        Validity
            Not Before: Mar 16 18:35:35 2026 GMT
            Not After : Mar 15 18:35:34 2028 GMT
        Subject: C = US, ST = Florida, L = Sunrise, O = Example Company, OU = IT Department, CN =
www.MyWebServer.com, emailAddress = info@MyWebServer.com
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (2048 bit)
                Modulus:
                    00:e3:27:b2:02:1c:01:28:81:45:37:0e:d3:c7:de:

                    ...

                    22:6a:09:80:39:b1:ac:dc:18:8c:50:32:59:d1:5e:
                    c3:8b
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints: critical
                CA:FALSE
            X509v3 Authority Key Identifier:
                98:51:FA:2D:B0:73:98:2B:F5:49:D9:D8:EE:90:3F:EF:7E:45:15:66
            X509v3 Subject Alternative Name:
                email:<username>@entrust.com, DNS:www.MyWebServer.com, DNS:MyWebServer.com
            X509v3 Extended Key Usage:
                TLS Web Server Authentication
            X509v3 Subject Key Identifier:
                8B:40:E5:B3:76:DD:A2:AA:96:00:50:E6:67:99:05:13:DF:9F:A2:F2
            X509v3 Key Usage: critical
                Digital Signature, Key Encipherment
```

```
...
```

Chapter 8. Additional resources and related products

8.1. nShield HSMs

8.2. nShield as a Service

8.3. Entrust products

8.4. nShield product documentation