Hyperledger Fabric

nShield® HSM Integration Guide

2024-10-21

Table of Contents

Chapter 1. Introduction

This document describes how to integrate Hyperledger Fabric with the Entrust nShield Container Option Pack (nCOP). The integration uses an Entrust nShield hardware security module as the root of trust for storage encryption to protect the private keys and meet FIPS 140 Level 2 and 3 criteria.

1.1. Product configurations

Entrust has successfully tested nShield HSM integration with Hyperledger Fabric in the following configurations:

Product	Version
Hyperledger Fabric	2.5
Security World	13.6.3
nCOP	1.1.2
Docker	27.3.1
Go	1.23.2
Host OS	Red Hat Enterprise Linux 9
Container OS	Ubuntu

1.2. Supported nShield hardware and software versions

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

1.2.1. Connect XC

Security World Software	Firmware	Image	OCS	Softcard	Module	FIPS Level 3
13.6.3	12.72.1 (FIPS 140-2 certified)	13.4.5	✓	✓	✓	✓

1.2.2. nShield 5C

Security World Software	Firmware	Image	OCS	Softcard	Module	FIPS Level 3
13.6.3	13.2.4 (FIPS 140-3 certified)	13.6.1	✓	✓	✓	✓

1.3. Supported nShield HSM functionality

Feature	Support
Module-only key	Yes
OCS cards	Yes
Softcards	Yes
nSaaS	Yes
FIPS 140 Level 3	Yes

1.4. Requirements

Familiarize yourself with:

- Hyperledger Fabric documentation: Hyperledger Fabric CA User's Guide.
- The nShield HSM: *Installation Guide* and *User Guide*.
- Your organizational Certificate Policy and Certificate Practice Statement, and

a Security Policy or Procedure in place covering administration of the PKI and HSM:

- The number and quorum of administrator cards in the Administrator Card Set (ACS), and the policy for managing these cards.
- The number and quorum of operator cards in the Operator Card Set (OCS), and the policy for managing these cards.
- The keys protection method: Module, Softcard, or OCS.
- The level of compliance for the Security World, FIPS 140 Level 3.
- Key attributes such as key size, time-out, or need for auditing key usage.

Entrust recommends that you allow only unprivileged connections unless you are performing administrative tasks.

Chapter 2. Procedures

2.1. Prerequisites:

1. Install the Entrust nShield HSM using the instructions in the *Installation Guide* for the HSM.
2. Configure the Entrust nShield HSM to have the IP address of your container host machine as a client.
3. Install the Entrust nShield Security World Software, and configure the Security World as described in the *User Guide* for the HSM. The Security World and module files will be copied to the container. Instructions for this are detailed later in this guide.
4. Edit or create the `cknfastrc` file located in `/opt/nfast/`:
    - If using Module protection:

      ```
      CKNFAST_FAKE_ACCELERATOR_LOGIN=1
      ```

    - If using OCS or Softcard protection:

      ```
      CKNFAST_NO_ACCELERATOR_SLOTS=1
      CKNFAST_LOADSHARING=1
      ```

5. Install Git, cURL, Go, and Docker. See the Hyperledger Fabric documentation for more information on the prerequisites.

2.2. Protection methods

2.2.1. Create OCS cardset for OCS protection.

If using OCS protection, make sure the OCS cardset has been created. If not using OCS protection and a FIPS Level 3 world file is used, the OCS card is still needed for FIPS authorization.

```
% createocs -m1 -s2 --persist -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: unformatted card
 Module 1 slot 3: empty
```

```
 Module 1 slot 4: empty
 Module 1 slot 5: empty
 Module 1 slot 2:- passphrase specified - writing card
Card writing complete.

cardset created; hkltu = c2ba9c6c4d169e4a2ca3908ca0a27832fcb0746e
```

You will need to have the OCS card mounted to be able to provide FIPS Authorization when using FIPS Level 3 world file.

2.2.2. Create a softcard for softcard protection

If using softcard protection, make sure the softcard has been created. If using FIPS Level 3 world file, an OCS cardset is still needed for FIPS authorization.

```
% ppmk -n testSC

Enter new pass phrase:
Enter new pass phrase again:
New softcard created: HKLTU bedcfb1a55bb706146770b0ad8180734aafb4dec
```

2.3. Download and install Hyperledger Fabric

1. Create a working directory for the installation:

   For example, Go Developers use the `$HOME/go/src/github.com/<your_github_userid>` directory. This is a Golang Community recommendation for Go projects. We will do the following:

```
% mkdir -p $HOME/go/src/github.com/install
% cd $HOME/go/src/github.com/install
```

1. Get the install script:

```
% curl -sSLO https://raw.githubusercontent.com/hyperledger/fabric/main/scripts/install-fabric.sh && chmod
+x install-fabric.sh
```

2. To install the Docker containers and binaries for Hyperledger Fabric, run the following command:

```
% ./install-fabric.sh docker binary

Pull Hyperledger Fabric binaries

===> Downloading version 2.5.9 platform specific fabric binaries
===> Downloading:  https://github.com/hyperledger/fabric/releases/download/v2.5.9/hyperledger-fabric-linux-
amd64-2.5.9.tar.gz
```

```
===> Will unpack to: /home/xyz/go/src/github.com/install
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100  111M  100  111M    0     0  23.2M      0  0:00:04  0:00:04 --:--:-- 27.6M
==> Done.
===> Downloading version 1.5.12 platform specific fabric-ca-client binary
===> Downloading:  https://github.com/hyperledger/fabric-ca/releases/download/v1.5.12/hyperledger-fabric-
ca-linux-amd64-1.5.12.tar.gz
===> Will unpack to: /home/xyz/go/src/github.com/install
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
  0     0    0     0    0     0      0      0 --:--:-- --:--:-- --:--:--     0
100 29.0M  100 29.0M    0     0  19.1M      0  0:00:01  0:00:01 --:--:-- 27.5M
==> Done.

Pull Hyperledger Fabric docker images

FABRIC_IMAGES: peer orderer ccenv baseos
===> Pulling fabric Images
.
.
.
Status: Downloaded newer image for hyperledger/fabric-ca:1.5.12
docker.io/hyperledger/fabric-ca:1.5.12
===> List out hyperledger images
hyperledger/fabric-peer      2.5       e9702d423cd4   3 months ago   142MB
hyperledger/fabric-peer      2.5.9     e9702d423cd4   3 months ago   142MB
hyperledger/fabric-peer      latest    e9702d423cd4   3 months ago   142MB
hyperledger/fabric-orderer   2.5       10fb520e9b0a   3 months ago   111MB
hyperledger/fabric-orderer   2.5.9     10fb520e9b0a   3 months ago   111MB
hyperledger/fabric-orderer   latest    10fb520e9b0a   3 months ago   111MB
hyperledger/fabric-ccenv     2.5       09ef9881ad5f   3 months ago   638MB
hyperledger/fabric-ccenv     2.5.9     09ef9881ad5f   3 months ago   638MB
hyperledger/fabric-ccenv     latest    09ef9881ad5f   3 months ago   638MB
hyperledger/fabric-baseos    2.5       b6e93e2f93f9   3 months ago   129MB
hyperledger/fabric-baseos    2.5.9     b6e93e2f93f9   3 months ago   129MB
hyperledger/fabric-baseos    latest    b6e93e2f93f9   3 months ago   129MB
hyperledger/fabric-ca        1.5       e324dcb92c6e   3 months ago   209MB
hyperledger/fabric-ca        1.5.12    e324dcb92c6e   3 months ago   209MB
hyperledger/fabric-ca        latest    e324dcb92c6e   3 months ago   209MB
```

2.4. Install nCOP on the host machine

The following is an example installation of nCOP. See *nShield Container Option Pack User Guide* for more details.

1. Log in to the container host machine server as **root**, and launch a terminal window.
2. Set up the nCOP working directory:

```
% mkdir -p /opt/ncop

% tar xf ncop-1.1.2.tar -C /opt/ncop
```

2.5. Setup and build nCOP image containers

We will use Ubuntu as the base operating system for the nCOP containers.

1. Change directory to where nCOP was installed:

```
% cd /opt/ncop
```

2. Create a mount directory:

```
% sudo mkdir /mnt/iso1
```

3. Mount the Security World Software ISO file:

```
% sudo mount -t iso9660 -o loop SecWorld_Lin64-13.6.3.iso /mnt/iso1
```

4. Build the nShield container for the hardserver (Ubuntu):

```
% sudo ./make-nshield-hwsp --tag nshield-hwsp-ubuntu:13.6.3 --from ubuntu /mnt/iso1
```

5. Build the Hyperledger nShield container for the Hyperledger application:

   This is the container application that we will use to configure Hyperledger Fabric. The `fabric-ca:1.5.12` container will be used as the base for the application OS. This Docker image was created when we installed Hyperledger Fabric. We will name the application `hyperpkcs11nshield:13.6.3`.

```
% sudo ./make-nshield-application --from hyperledger/fabric-ca:1.5.12 --tag hyperpkcs11nshield:13.6.3
/mnt/iso1


Detecting nShield software version
Version is 13.6.3
Unpacking /mnt/iso1/linux/amd64/hwsp.tar.gz ...
Unpacking /mnt/iso1/linux/amd64/ctls.tar.gz ...
Adding files...
Building image...
[+] Building 0.5s (16/16) FINISHED
docker:default
 => [internal] load build definition from Dockerfile
0.0s
 => => transferring dockerfile: 1.36kB
0.0s
 => [internal] load metadata for docker.io/hyperledger/fabric-ca:1.5.12
0.0s
 => [internal] load .dockerignore
0.0s
 => => transferring context: 2B
0.0s
 => [ 1/11] FROM docker.io/hyperledger/fabric-ca:1.5.12
0.0s
 => [internal] load build context
0.5s
 => => transferring context: 1.93MB
```

```
0.4s
 => CACHED [ 2/11] RUN grep "release 6" /etc/redhat-release >/dev/null 2>&1; if [ "$?" -eq "0" ]; then yum
-y install epel-release; fi                                                                    0.0s
 => CACHED [ 3/11] RUN if [ -x /usr/bin/microdnf ]; then microdnf update && microdnf install socat &&
microdnf clean all; fi                                                                         0.0s
 => CACHED [ 4/11] RUN grep "release 8" /etc/redhat-release >/dev/null 2>&1; if [ "$?" -eq "0" ] && [ -x
/usr/bin/microdnf ] ; then microdnf install libnsl2; fi #no libnsl2 in UBI8-minimal  0.0s
 => CACHED [ 5/11] RUN if [ -x /usr/bin/yum ]; then yum -y update && yum -y install socat libnsl2 && yum
clean all; fi                                                                                  0.0s
 => CACHED [ 6/11] RUN grep "release 8" /etc/redhat-release >/dev/null 2>&1; if [ "$?" -eq "0" ]; then ln
-s /usr/lib64/libnsl.so.2 /usr/lib64/libnsl.so.1; fi #no libnsl in UBI                         0.0s
 => CACHED [ 7/11] RUN if [ -x /usr/bin/apt-get ]; then apt-get -y update && apt-get -y upgrade && apt-get
-y install socat; fi                                                                           0.0s
 => CACHED [ 8/11] RUN if [ -x /usr/bin/zypper ]; then zypper update -y && zypper install -y socat &&
zypper clean --all; fi                                                                         0.0s
 => CACHED [ 9/11] RUN if [ -x /sbin/apk ]; then apk update && apk add socat && apk add bash ; fi
0.0s
 => CACHED [10/11] COPY opt /opt
0.0s
 => CACHED [11/11] RUN mkdir -p /opt/nfast/kmdata /opt/nfast/sockets && mkdir -m 1755 /opt/nfast/kmdata/tmp
0.0s
 => exporting to image
0.0s
 => => exporting layers
0.0s
 => => writing image sha256:f12ef9b636842c646146ba7d5848f170abd380c9d82564126aebe75c0f437cd7
0.0s
 => => naming to docker.io/library/hyperpkcs11nshield:13.6.3
```

6. Validate the images have been built:

```
% docker images

REPOSITORY            TAG       IMAGE ID       CREATED         SIZE
hyperpkcs11nshield    13.6.3    f12ef9b63684   2 minutes ago   996MB
nshield-hwsp-ubuntu   13.6.3    dd72174f64a4   6 minutes ago   653MB
```

7. Configure `nshield-hwsp`:

   a. Set up the hardserver configuration file and directory:

```
% cd /opt/ncop

% mkdir -p /opt/ncop/config1

% ./make-nshield-hwsp-config --output /opt/ncop/config1/config <hsm ip address>

% cat /opt/ncop/config1/config
```

   b. Build the nShield Application Container Security World:

```
% sudo mkdir -p /opt/ncop/app1/kmdata/local

% sudo cp /opt/nfast/kmdata/local/world /opt/ncop/app1/kmdata/local/.

% sudo cp /opt/nfast/kmdata/local/module_* /opt/ncop/app1/kmdata/local/.

# Copy OCS cardset files
% sudo cp /opt/nfast/kmdata/local/card* /opt/ncop/app1/kmdata/local/.
```

```
# Copy Softcard files
% sudo cp /opt/nfast/kmdata/local/softcard* /opt/ncop/app1/kmdata/local/.

% ls /opt/ncop/app1/kmdata/*
```

c. Create a new socket so that application containers can use the hardserver:

```
% docker volume create socket1
```

8. Run the `nshield-hwsp` container:

```
% sudo docker run -v /opt/ncop/config1:/opt/nfast/kmdata/config:ro -v socket1:/opt/nfast/sockets nshield-
hwsp-ubuntu:13.6.3
```

a. Check the status of `nshield-hwsp` using the `enquiry` command:

```
% dmountpoint=`sudo docker volume inspect --format '{{ .Mountpoint }}' socket1`
% export NFAST_SERVER=$dmountpoint/nserver
% sudo /opt/nfast/bin/enquiry -m0

Server:
 enquiry reply flags  none
 enquiry reply level  Six
 serial number        7852-268D-3BF9
 mode                 operational
 version              13.6.3
 speed index          20000
 rec. queue           514..812
 level one flags      Hardware HasTokens SupportsCommandState
 version string       13.6.3-90-86c7a396, 13.2.4-280-7f4f0c24, 13.6.1-61-6acd63f8
 checked in           000000006671e78b Tue Jun 18 16:01:15 2024
 level two flags      none
 max. write size      8192
 level three flags    KeyStorage
 level four flags     HasRTC HasNVRAM HasNSOPermsCmd ServerHasPollCmds FastPollSlotList HasShareACL
HasFeatureEnable HasFileOp HasLongJobs ServerHasLongJobs AESModuleKeys NTokenCmds Type2Smartcard
ServerHasCreateClient HasInitialiseUnitEx AlwaysUseStrongPrim\
es Type3Smartcard HasKLF2
 module type code     0
 product name         nFast server
 device name
 EnquirySix version   8
 impath kx groups     DHPrime1024 DHPrime3072 DHPrime3072Ex DHPrimeMODP3072 DHPrimeMODP3072mGCM
 feature ctrl flags   none
 features enabled     none
 version serial       0
 level six flags      none
 remote port (IPv4)   9004
 kneti hash           b29d608519eb0549853ffa3573bf199dc58f91d8
 rec. LongJobs queue  0
 SEE machine type     None
 supported KML types
 active modes         none
 remote port (IPv6)   9004
```

9. Check the Hyperledger nShield Container Application using `enquiry`:

```
% docker run --rm -it -v /opt/ncop/app1/kmdata:/opt/nfast/kmdata:ro -v socket1:/opt/nfast/sockets -it
hyperpkcs11nshield:13.6.3 /opt/nfast/bin/enquiry -m0

Server:
 enquiry reply flags  none
 enquiry reply level  Six
 serial number        6A74-0000-7777
 mode                 operational
 version              13.6.3
 speed index          20000
 rec. queue           514..812
 level one flags      Hardware HasTokens SupportsCommandState
 version string       13.6.3-90-86c7a396, 13.2.4-280-7f4f0c24, 13.6.1-61-6acd63f8
 checked in           000000006671e78b Tue Jun 18 20:01:15 2024
 level two flags      none
 max. write size      8192
 level three flags    KeyStorage
 level four flags     HasRTC HasNVRAM HasNSOPermsCmd ServerHasPollCmds FastPollSlotList HasShareACL
HasFeatureEnable HasFileOp HasLongJobs ServerHasLongJobs AESModuleKeys NTokenCmds Type2Smartcard
ServerHasCreateClient HasInitialiseUnitEx Al
waysUseStrongPrimes Type3Smartcard HasKLF2
 module type code     0
 product name         nFast server
 device name
 EnquirySix version   8
 impath kx groups     DHPrime1024 DHPrime3072 DHPrime3072Ex DHPrimeMODP3072 DHPrimeMODP3072mGCM
 feature ctrl flags   none
 features enabled     none
 version serial       0
 level six flags      none
 remote port (IPv4)   9004
 kneti hash           cdaed24111d4607678759c1c58d652154734afac
 rec. LongJobs queue  0
 SEE machine type     None
 supported KML types
 active modes         none
 remote port (IPv6)   9004

Module #1:
 enquiry reply flags  UnprivOnly
 enquiry reply level  Six
 serial number        6A74-0000-7777
 mode                 operational
 version              13.2.4
 speed index          20000
 rec. queue           120..250
 level one flags      Hardware HasTokens SupportsCommandState SupportsHotReset
 version string       13.2.4-280-7f4f0c24, 13.6.1-61-6acd63f8
 checked in           00000000651fceee Fri Oct  6 09:10:06 2023
 level two flags      none
 max. write size      262152
 level three flags    KeyStorage
 level four flags     HasRTC HasNVRAM HasNSOPermsCmd ServerHasPollCmds FastPollSlotList HasShareACL
HasFeatureEnable HasFileOp HasLongJobs ServerHasLongJobs AESModuleKeys NTokenCmds Type2Smartcard
ServerHasCreateClient HasInitialiseUnitEx Al
waysUseStrongPrimes Type3Smartcard HasKLF2
 module type code     14
 product name         NH2096-0F
 device name          Rt1
 EnquirySix version   7
 impath kx groups     DHPrime1024 DHPrime3072 DHPrime3072Ex DHPrimeMODP3072
 feature ctrl flags   LongTerm
 features enabled     ForeignTokenOpen RemoteShare KISAAlgorithms StandardKM EllipticCurve ECCMQV
AcceleratedECC HSMSpeed2
 version serial       0
 connection status    OK
```

```
 connection info      esn = 6A74-1111-2222; addr = INET/10.000.000.00/9004; ku hash =
1bbf65966c0d03968ed02614303cc1b25c1f3a3a, mech = Any
 image version        13.6.1-50-6acd63f8
 level six flags      SerialConsoleAvailable Type3SmartcardRevB
 max exported modules 100
 rec. LongJobs queue  36
 SEE machine type     None
 supported KML types  DSAp1024s160 DSAp3072s256
 using impath kx grp  DHPrimeMODP3072mGCM
 active modes         UseFIPSApprovedInternalMechanisms AlwaysUseStrongPrimes
 physical serial      46-UUUUUU
 hardware part no     PCA10005-01 revision 03
 hardware status      OK
```

2.6. BCCSP fabric configuration

During the hyperledger configuration you will be asked to modify the `bccsp` section of different configuration files. Use the following configuration according to the protection method being used:

1. Module protection

```
bccsp:
    default: PKCS11
    pkcs11:
        Library: /opt/nfast/toolkits/pkcs11/libcknfast.so
        Pin: ncipher
        label: loadshared accelerator
        hash: SHA2
        security: 256
```

2. Softcard protection

```
bccsp:
    default: PKCS11
    pkcs11:
        Library: /opt/nfast/toolkits/pkcs11/libcknfast.so
        Pin: ncipher
        label: testSC
        hash: SHA2
        security: 256
```

3. OCS protection

```
bccsp:
    default: PKCS11
    pkcs11:
        Library: /opt/nfast/toolkits/pkcs11/libcknfast.so
        Pin: ncipher
        label: testOCS
        hash: SHA2
        security: 256
```

1. The label is the name of the softcard or OCS it will look for.
2. For module protection make the label: `loadshared accelerator`
3. If `CKNFAST_LOADSHARING` is not in `cknfastrc`, the label will just be `accelerator`.
4. Pin is not needed for module protection so it can be anything.

2.7. Configure and start the Hyperledger Fabric CA server

The volumes that will be used in the `docker run` command are:

-v /opt/ncop/app1/kmdata:/opt/nfast/kmdata:rw

Used to share `kmdata\local` between Security World and the app container.

-v socket1:/opt/nfast/sockets

Required for hardserver communication.

-v /opt/nfast/cknfastrc:/opt/nfast/cknfastrc

File used when selecting PKCS #11 key protection type.

-v /opt/nfast/kmdata/config/cardlist:/opt/nfast/kmdata/config/cardlist

File used to expose the cards available to the HSM.

-v /opt/nfast/toolkits/pkcs11/libcknfast.so:/opt/nfast/toolkits/pkcs11/libcknfast.so

PKCS11 library to be used by the hyperledger application.

1. Run the container interactively with a bash prompt:

```
docker run  -it \
     -v /opt/ncop/app1/kmdata:/opt/nfast/kmdata:rw \
     -v socket1:/opt/nfast/sockets \
     -v /opt/nfast/cknfastrc:/opt/nfast/cknfastrc \
     -v /opt/nfast/kmdata/config/cardlist:/opt/nfast/kmdata/config/cardlist \
     -v /opt/nfast/toolkits/pkcs11/libcknfast.so:/opt/nfast/toolkits/pkcs11/libcknfast.so \
     hyperpkcs11nshield:13.6.3
```

2. Run `enquiry` and `nfkminfo` within the container.

   The output should show a usable module and Security World.

```
% /opt/nfast/bin/enquiry

% /opt/nfast/bin/nfkminfo
```

3. Run `fabric-ca-server` to generate a new config file to edit.

   In this step, `fabric-ca-server-config.yaml` will be generated and then edited within the container.

```
% fabric-ca-server start -b root:root

2024/10/07 19:41:00 [INFO] Created default configuration file at /etc/hyperledger/fabric-ca-
server/fabric-ca-server-config.yaml
2024/10/07 19:41:00 [INFO] Starting server in home directory: /etc/hyperledger/fabric-ca-server
2024/10/07 19:41:00 [INFO] Server Version: v1.5.12
2024/10/07 19:41:00 [INFO] Server Levels: &{Identity:2 Affiliation:1 Certificate:1 Credential:1
RAInfo:1 Nonce:1}
2024/10/07 19:41:00 [WARNING] &{69 The specified CA certificate file /etc/hyperledger/fabric-ca-
server/ca-cert.pem does not exist}
2024/10/07 19:41:00 [INFO] generating key: &{A:ecdsa S:256}
2024/10/07 19:41:00 [INFO] encoded CSR
2024/10/07 19:41:00 [INFO] signed certificate with serial number
117486704313834000000000004842645971111111111116
2024/10/07 19:41:00 [INFO] The CA key and certificate were generated for CA
2024/10/07 19:41:00 [INFO] The key was stored by BCCSP provider 'SW'
2024/10/07 19:41:00 [INFO] The certificate is at: /etc/hyperledger/fabric-ca-server/ca-cert.pem
2024/10/07 19:41:00 [INFO] Initialized sqlite3 database at /etc/hyperledger/fabric-ca-server/fabric-
ca-server.db
2024/10/07 19:41:00 [INFO] The issuer key was successfully stored. The public key is at:
/etc/hyperledger/fabric-ca-server/IssuerPublicKey, secret key is at: /etc/hyperledger/fabric-ca-
server/msp/keystore/IssuerSecretKey
2024/10/07 19:41:00 [INFO] Idemix issuer revocation public and secret keys were generated for CA ''
2024/10/07 19:41:00 [INFO] The revocation key was successfully stored. The public key is at:
/etc/hyperledger/fabric-ca-server/IssuerRevocationPublicKey, private key is at:
/etc/hyperledger/fabric-ca-server/msp/keystore/IssuerRevocationPrivateKey
2024/10/07 19:41:00 [INFO] Home directory for default CA: /etc/hyperledger/fabric-ca-server
2024/10/07 19:41:00 [INFO] Operation Server Listening on 127.0.0.1:9443
2024/10/07 19:41:00 [INFO] Listening on http://0.0.0.0:7054
```

   

   Type **Control-C** or **X** multiple times to exit.

4. Install the vi editor in the container.

```
% apt-get install vim -y
```

5. Edit the `fabric-ca-server-config.yaml` file.

   This file was created in the `/etc/hyperledger/fabric-ca-server` directory.

```
% cd /etc/hyperledger/fabric-ca-server

% vi fabric-ca-server-config.yaml
```

   a. Find the "bccsp" section and add the PKCS #11 settings as discussed in the BCCSP fabric configuration section. For example:

```
bccsp:
    default: PKCS11
```

```
    pkcs11:
        Library: /opt/nfast/toolkits/pkcs11/libcknfast.so
        Pin: 123456
        label: fabric
        hash: SHA2
        security: 256
```

In this example:

- The name of the Softcard or OCS is `fabric` and the pin is `123456`.
- If using module protection, the label will be `accelerator`.
- If using module protection with `CKNFAST_LOADSHARING=1` in the `/opt/nfast/cknfastrc` file, the label will be `loadshared accelerator`.
- The pin can be anything if using module protection.

b. Save the file.

6. Delete any keystore in `/etc/hyperledger/fabric-ca-server`, such as the `msp` directory and the old files so that new ones are generated with the HSM when the server is started.
   a. Delete any keystore such as msp/ directory (`/etc/hyperledger/fabric-ca-server/msp`).
   b. Delete ca-cert.pem it created before ()`/etc/hyperledger/fabric-ca-server/ca-cert.pem`).
   c. Delete `IssuerPublicKey` and `IssuerRevocationPublicKey` in `/etc/hyperledger/fabric-ca-server`.
7. Start the server: Now when the server starts it has our `pkcs11 bccsp` profile and no existing key or cert in the hyperledger directory.

```
% fabric-ca-server start

2024/10/07 19:57:51 [INFO] Configuration file location: /etc/hyperledger/fabric-ca-server/fabric-ca-
server-config.yaml
2024/10/07 19:57:51 [INFO] Starting server in home directory: /etc/hyperledger/fabric-ca-server
2024/10/07 19:57:51 [DEBUG] Set log level:
2024/10/07 19:57:51 [INFO] Server Version: v1.5.12
2024/10/07 19:57:51 [INFO] Server Levels: &{Identity:2 Affiliation:1 Certificate:1 Credential:1
RAInfo:1 Nonce:1}
2024/10/07 19:57:51 [DEBUG] Making server filenames absolute
2024/10/07 19:57:51 [DEBUG] Initializing default CA in directory /etc/hyperledger/fabric-ca-server
.
.
.
2024/10/07 19:57:51 [INFO] generating key: &{A:ecdsa S:256}
2024/10/07 19:57:51 [DEBUG] generate key from request: algo=ecdsa, size=256
2024-10-07 19:57:51.683 UTC 0001 INFO [bccsp_p11] generateECKey -> Generated new P11 key, SKI
505dcfa5deb97cfbd0f0b1c8c59fac50d3442ef97387cc7bf91fe5cfcf31f1cc
.
.
.
2024/10/07 19:57:51 [INFO] signed certificate with serial number
682578444444444400969769333333333216663525563
```

```
2024/10/07 19:57:51 [INFO] The CA key and certificate were generated for CA
2024/10/07 19:57:51 [INFO] The key was stored by BCCSP provider 'PKCS11'
2024/10/07 19:57:51 [INFO] The certificate is at: /etc/hyperledger/fabric-ca-server/ca-cert.pem
.
.
.
2024/10/07 19:57:51 [INFO] Home directory for default CA: /etc/hyperledger/fabric-ca-server
2024/10/07 19:57:51 [DEBUG] 1 CA instance(s) running on server
2024/10/07 19:57:51 [INFO] Operation Server Listening on 127.0.0.1:9443
2024/10/07 19:57:51 [INFO] Listening on http://0.0.0.0:7054
```

Now the pkcs11 initialization stage is complete as shown in the output.

The new HSM protected PKCS #11 key can be found at `/opt/nfast/kmdata/local`. The cert is in the Hyperledger directory as `/etc/hyperledger/fabric-ca-server/ca-cert.pem`.

You will need to have the OCS card available to be able to provide FIPS Authorization when using FIPS Level 3 world file.

8. Test to see if the key was created.

   In another window go inside the container that is running the fabric-ca-server. You can do this by listing the docker containers that are running.

```
% docker ps
```

   Now run the following command to enter the container.

```
% docker exec -i -t <container id from docker ps> /bin/bash
```

   Inside the container list the contents of the `/opt/nfast/kmdata/local`. You should see the key file there.

```
% ls -al /opt/nfast/kmdata/local

total 60
drwxr-xr-x. 2 root root   109 Oct  7 19:57 .
drwxr-xr-x. 4 root root    33 Oct  7 17:31 ..
-rw-r--r--. 1 root root  8208 Oct  7 19:57 key_pkcs11_ua15uuuuuuuuucd6498b4ca9492cbf9a356cf94d1
-rwxr-xr-x. 1 root root  5204 Oct  7 18:13 module_6A74-1261-7843
-rwxr-xr-x. 1 root root 37752 Oct  7 18:13 world
```

   Run nfkminfo to list the keys.

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

```
Keys with softcard protection:
 key_pkcs11_ua15uuuuuuuuuucd6498b4ca9492cbf9a356cf94d1
`505dcfa5deb97cfbd0f0b1aaaaaaaaaaef97387cc7bf91fe5cfcf31f1cc'
```

2.8. Enroll and register a Fabric CA client

1. Edit the `fabric-ca-server-config.yaml` file and change the identity section as needed.
2. Start the `fabric-ca-server` if it is not currently running. It needs to be running for the `enroll` command to work.
3. In a separate window, go inside the container that is running the `fabric-ca-server` as described before.
4. Create the `FABRIC_CA_CLIENT_HOME` environment variable:

```
% export FABRIC_CA_CLIENT_HOME=$HOME/fabric-ca/clients/admin
```

5. Enroll using the identity in the server YAML file: In this example is `root/root`.

```
% fabric-ca-client enroll -u http://root:root@localhost:7054

2024/10/08 18:09:13 [INFO] Created a default configuration file at /root/fabric-ca/clients/admin/fabric-ca-client-config.yaml
2024/10/08 18:09:13 [INFO] generating key: &{A:ecdsa S:256}
2024/10/08 18:09:13 [INFO] encoded CSR
2024/10/08 18:09:13 [INFO] Stored client certificate at /root/fabric-ca/clients/admin/msp/signcerts/cert.pem
2024/10/08 18:09:13 [INFO] Stored root CA certificate at /root/fabric-ca/clients/admin/msp/cacerts/localhost-7054.pem
2024/10/08 18:09:13 [INFO] Stored Issuer public key at /root/fabric-ca/clients/admin/msp/IssuerPublicKey
2024/10/08 18:09:13 [INFO] Stored Issuer revocation public key at /root/fabric-ca/clients/admin/msp/IssuerRevocationPublicKey
```

   This generates a client YAML file to edit. The client is not yet enrolled through the HSM.

6. Edit the client YAML file in `$HOME/fabric-ca/clients/admin` folder.
   a. Edit the "bccsp" section and mirror the server YAML BCCSP for the HSM.
   b. Save the file.
7. Delete the `msp` directory in `$HOME/fabric-ca/clients/admin` folder.

```
% rm -rf $HOME/fabric-ca/clients/admin/msp
```

8. Run the `enroll` command again with the server identity:

```
% fabric-ca-client enroll -u http://root:root@localhost:7054
```

```
2024/10/08 18:36:17 [INFO] generating key: &{A:ecdsa S:256}
2024-10-08 18:36:18.039 UTC 0001 INFO [bccsp_p11] generateECKey -> Generated new P11 key, SKI
0783bb7fe1cb8439365aebbbbbbbbbb3cedcff3c9879aae9d157a57fa8a2
2024/10/08 18:36:18 [INFO] encoded CSR
2024/10/08 18:36:18 [INFO] Stored client certificate at /root/fabric-
ca/clients/admin/msp/signcerts/cert.pem
2024/10/08 18:36:18 [INFO] Stored root CA certificate at /root/fabric-
ca/clients/admin/msp/cacerts/localhost-7054.pem
2024/10/08 18:36:18 [INFO] Stored Issuer public key at /root/fabric-ca/clients/admin/msp/IssuerPublicKey
2024/10/08 18:36:18 [INFO] Stored Issuer revocation public key at /root/fabric-
ca/clients/admin/msp/IssuerRevocationPublicKey
```

You will need to have the OCS card available to be able to provide FIPS Authorization when using FIPS Level 3 world file. You can see in the output that a new key was generated.

9. Register the client. For example:

```
% fabric-ca-client register --id.name ica.example --id.type client --id.secret root --csr.names
C=es,ST=madrid,L=Madrid,O=example.com --csr.cn ica.example -m ica.example --id.attrs
'"hf.IntermediateCA=true"' -u http://localhost:7054 --loglevel debug
```

2.9. Peers and ordering nodes

To set up peers and ordering nodes with the Entrust nShield HSM:

1. Edit one more YAML file for each node.
   a. For the PKCS #11 BCCSP template, mirror the server YAML BCCSP for the HSM.
   b. This will be the `core.yaml` file for a peer node and the `orderer.yaml` file for a ordering node.
2. Run the enrollment lines from the peer or ordering nodes to the main fabric CA server to enroll it.

See the Hyperledger Fabric documentation for more information.

Chapter 3. Additional resources and related products

3.1. nShield Connect

3.2. nShield as a Service

3.3. nShield Container Option Pack

3.4. Entrust products

3.5. nShield product documentation