HashiCorp Vault Enterprise

nShield® HSM Integration Guide

11 Apr 2023

Contents

1. Introduction

HashiCorp Vault Enterprise (referred to as Vault in this guide) supports the creation/storage of keys within Hardware Security Modules (HSMs). Entrust nShield HSMs provide FIPS or Common Criteria certified solutions to securely generate, encrypt, and decrypt the keys which provide the root of trust for the Vault protection mechanism.

This guide describes how to integrate Vault with an nShield HSM to:

- Offload select PKI operations to the HSMs.
- Generate new PKI key pairs and certificates.
- Verify and sign certificate workflows.

1.1. Product configurations

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

Product	Version
HashiCorp Vault Enterprise	v1.10.0 Enterprise HSM
Base OS	Red Hat Enterprise 8.4

1.1.1. 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

1.1.2. Supported nShield hardware and software versions

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

1.1.2.1. nShield 5c

Security World Software	Firmware	Netimage	OCS	Softcard	Module
13.2.2	13.2.2 (FIPS Pending)	13.2.2	✓	✓	✓

1.1.2.2. Connect XC

Security World Software	Firmware	Netimage	OCS	Softcard	Module
12.80.4	12.50.11 (FIPS Certified)	12.80.4	✓	✓	✓
12.80.4	12.60.15 (CC Certified)	12.80.4	✓	✓	✓

1.1.2.3. Connect +

Security World Software	Firmware	Netimage	OCS	Softcard	Module
12.80.4	12.50.8 (FIPS Certified)	12.80.4	✓	✓	✓
12.40 Compatibility Package	2.55.4 (CC Certified)	12.45.1	✓	✓	✓

1.1.3. Supported nShield key types

Entrust has successfully tested with the following Vault managed keys:

- RSA
- ECDSA

1.2. Requirements

Before installing these products, read the associated nShield HSM *Installation Guide*, *User Guide*, and the Vault documentation. This guide assumes familiarity with the following:

- 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 140-2 Level 3 or Common Criteria standards. If using FIPS Restricted mode, it is advisable to create an OCS for FIPS authorization. The OCS can also provide key protection for the Vault master key. For information about limitations on FIPS authorization, see the *Installation Guide* of the nShield HSM.
- Whether to instantiate the Security World as recoverable or not.
- Network environment setup, via correct firewall configuration with usable ports: 9004 for the HSM and 8200 for Vault.
- HashiCorp Vault Enterprise Modules license, which is required for using Vault with Hardware Security Modules.

1.3. More information

For more information about OS support, contact your HashiCorp Vault Enterprise sales representative or Entrust nShield Support, https://nshieldsupport.entrust.com.

2. Integration procedures

A dedicated Linux server is needed for the installation of the Vault.

Follow these steps to install and configure the Vault with a single HSM:

1. System preparation
2. Install the HSM
3. Install the Security World software and create a Security World
4. Generate the Vault encryption and HMAC keys with OCS and Softcard protection
5. Verify the PKCS#11 library is available
6. Find the slot value for each protection method
7. Create Vault user and group
8. Install Vault
9. Install the Vault license
10. Create a configuration file
11. Create and configure Vault directories
12. Enable Vault
13. Start Vault
14. Log in from the command line
15. Create Managed Key In Vault

2.1. System preparation

1. Open the appropriate firewall port for incoming HSM connections:

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

2. Open the appropriate firewall port for incoming Vault connections:

```
# sudo firewall-cmd --permanent --add-port=8200/tcp
# sudo firewall-cmd --permanent --add-port=8201/tcp
```

3. Apply the firewall changes above:

```
# sudo firewall-cmd --reload
```

4. Install `open-vm-tools`:

```
# sudo yum install open-vm-tools unzip opensc
```

2.2. Install the HSM

Install the nShield Connect HSM locally, remotely, or remotely via the serial console. See the following nShield Support articles, and the *Installation Guide* for the HSM:

- https://nshieldsupport.entrust.com/hc/en-us/articles/360021378272-How-To-Locally-Set-up-a-new-or-replacement-nShield-Connect
- https://nshieldsupport.entrust.com/hc/en-us/articles/360014011798-How-To-Remotely-Setup-a-new-or-replacement-nShield-Connect
- https://nshieldsupport.entrust.com/hc/en-us/articles/360013253417-How-To-Remotely-Setup-a-new-or-replacement-nShield-Connect-XC-Serial-Console-Model

2.3. Install the Security World software and create a Security World

1. Install and configure the Security World software. For instructions, see the *Installation Guide* and the *User Guide* for the HSM.
2. Install the TAC-955 hot fix. This hotfix contains an updated version of the PKCS#11 library and utilities.
3. Add `$NFAST_HOME` on the path variable:

   ```
   # sudo vi /etc/profile.d/nfast.sh
   ```

   Add the following info to `nfast.sh` and save:

   ```
   # Entrust Security World path variable
   export PATH=$PATH:/opt/nfast/bin
   ```

4. Restart the server.
5. Confirm that the HSM is available:

   ```
   # enquiry
   Server:
    enquiry reply flags  none
    enquiry reply level  Six
    serial number        C6BB-CAAF-ADBF
    mode                 operational
    ...
   ```

6. Create your Security World if one does not already exist. Follow your organization's security policy for this. Create extra ACS cards, one for each person with access privilege, plus spares.

After an ACS card set has been created, the cards cannot be duplicated.

7. Confirm that the Security World is **operational** and **usable** in the output of the `nfkminfo` command:

```
# nfkminfo
World
 generation  2
 state       0x3737000c Initialised Usable Recovery !PINRecovery !ExistingClient RTC NVRAM FTO AlwaysUseStrongPrimes
!DisablePKCS1Padding !PpStrengthCheck !AuditLogging SEEDebug AdminAuthRequired
 n_modules   1
 hknso       2f9718603b4cdb9d4f30913dfa340ffea71aa576
 hkm         9d87b5036f724f02378d3b2e2e643b9a3f03839a (type Rijndael)
 hkmwk       c2be99fe1c77f1b75d48e2fd2df8dffc0c969bcb
 hkre        78a5664bdd655bfbe84477ba05440b3fbdaa6a07
 hkra        a3cc2e2a9129f6d38b288dc09b9b9d4768a4d34b
 hkfips      5ab63d2f8b50c3d5485ee1d68f1bcaa74da63390
 hkmc        ec1385f4e37bba2f00ba009b3b4708de1d430048
 hkrtc       39422cec6d716527f2e9670ed8281b636f2599ec
 hknv        5c63f4d1c0ecd206bcc8fd31dedd29560958c598
 hkdsee      ebd5e839fae5c277fcf535eca3022b8d55d95c77
 hkfto       f485795962116cd04d85447058599c6f98ad81db
 hkmnull     0100000000000000000000000000000000000000
 ex.client   none
 k-out-of-n  1/2
 other quora m=1 r=1 nv=1 rtc=1 dsee=1 fto=1
 createtime  2022-02-04 19:08:39
 nso timeout 10 min
 ciphersuite DLf3072s256mAEScSP800131Ar1
 min pp      0 chars
 mode        fips1402level3
 ...
```

2.4. Generate the Vault encryption and HMAC keys with OCS and Softcard protection

The Vault encryption and HMAC keys can be protected with an OCS, Softcard or Module:

- Operator Cards Set (OCS) are smartcards that are presented to the physical smartcard reader of an HSM. If an OCS is used, k must = 1 whereas N can be up to, but not exceed, 64. For more information on OCS use, properties, and k-of-N values, see the *User Guide* for your HSM.
- Softcards are logical tokens (passphrases) that protect they key and authorize its use.
- Module protection has no passphrase.

Key generation with all three protection methods are shown below. Choose those that apply to you.

2.4.1. Generate the keys using an OCS protection

1. Create the OCS. Follow your organization's security policy for the values of K/N. Create extra OCS cards, one for each person with access privilege, plus spares.

Administrator Card Set (ACS) authorization is required to create an OCS in FIPS 140-2 level 3.

After an OCS card set has been created, the cards cannot be duplicated.

```
# createocs -m1 -s2 -N HashiCorpOCS -Q 1/1

FIPS 140-2 level 3 auth obtained.

Creating Cardset:
 Module 1: 0 cards of 1 written
 Module 1 slot 3: Admin Card #1
 Module 1 slot 2: blank card
 Module 1 slot 0: empty
 Module 1 slot 2:- passphrase specified - writing card
Card writing complete.

cardset created; hkltu = 5ee369387f1bfa077f9186faaea25b50f1b668dc
```

2. List the OCS created.

```
# nfkminfo -c
Cardset list - 1 cardsets:  (P)ersistent/(N)ot, (R)emoteable/(L)ocal-only
 Operator logical token hash              k/n timeout  name
 5ee369387f1bfa077f9186faaea25b50f1b668dc  1/1  none-NL HashiCorpOCS
```

3. Create the Vault encryption key `vault_v1_ocs`.

```
# generatekey --generate --batch -m1 -s2 pkcs11 protect=token cardset=HashiCorpOCS plainname=vault_v1_ocs type=AES
size=256
key generation parameters:
 operation    Operation to perform       generate
 application  Application                pkcs11
 protect      Protected by               token
 slot         Slot to read cards from    2
 recovery     Key recovery               yes
 verify       Verify security of key     yes
 type         Key type                   AES
 size         Key size                   256
 plainname    Key name                   vault_v1_ocs
 nvram        Blob in NVRAM (needs ACS)  no

Loading `HashiCorpOCS':
 Module 1: 0 cards of 1 read
 Module 1 slot 2: `HashiCorpOCS' #1
 Module 1 slot 3: Admin Card #1
 Module 1 slot 0: empty
 Module 1 slot 2:- passphrase supplied - reading card
Card reading complete.

Key successfully generated.
Path to key: /opt/nfast/kmdata/local/key_pkcs11_uc5ee369387f1bfa077f9186faaea25b50f1b668dc-
5f2a530b5da1ba75861446871e14860e37faf3d3
```

4. Create the Vault HMAC key `vault_hmac_v1_ocs`.

```
#  generatekey --generate --batch -m1 -s2 pkcs11 protect=token cardset=HashiCorpOCS plainname=vault_hmac_v1_ocs
type=HMACSHA256 size=256
key generation parameters:
 operation    Operation to perform       generate
 application  Application                pkcs11
 protect      Protected by               token
 slot         Slot to read cards from    2
 recovery     Key recovery               yes
 verify       Verify security of key     yes
 type         Key type                   HMACSHA256
 size         Key size                   256
 plainname    Key name                   vault_hmac_v1_ocs
 nvram        Blob in NVRAM (needs ACS)  no

Loading `HashiCorpOCS':
 Module 1: 0 cards of 1 read
 Module 1 slot 2: `HashiCorpOCS' #1
 Module 1 slot 3: Admin Card #1
 Module 1 slot 0: empty
 Module 1 slot 2:- passphrase supplied - reading card
Card reading complete.

Key successfully generated.
Path to key: /opt/nfast/kmdata/local/key_pkcs11_uc5ee369387f1bfa077f9186faaea25b50f1b668dc-
fac560f20f3a5d34ccfd3d288427f57cf3f24cd4
```

2.4.2. Generate the keys using Softcard and Module protection

1. Create a `/opt/nfast/cknfastrc` file with the following content:

```
# cat /opt/nfast/cknfastrc
CKNFAST_LOADSHARING=1
CKNFAST_FAKE_ACCELERATOR_LOGIN=1
CKNFAST_DEBUG=10
CKNFAST_DEBUGFILE=/opt/nfast/log/pkcs11.log
```

The first line enables Softcard protection support. The second enables Module protection support. Third and fourth enable PKCS11 log files which you will need to find the `slot` to configure the Vault.

2. Create the Softcard token using the `ppmk` command. Enter a passphrase or password at the prompt.

```
# ppmk -n HashiCorpSC

Enter new pass phrase:
Enter new pass phrase again:
New softcard created: HKLTU 702bec9b07c5a289ece6aaff880931b5003c8acd
```

3. List the Softcard created.

```
# nfkminfo -s
SoftCard summary - 1 softcards:
 Operator logical token hash               name
 702bec9b07c5a289ece6aaff880931b5003c8acd  HashiCorpSC
```

4. Create an encryption key `vault_v1_sc` using Softcard protection:

```
# generatekey --generate --batch -m1 pkcs11 protect=softcard softcard=HashiCorpSC plainname=vault_v1_sc type=AES
size=256
key generation parameters:
 operation    Operation to perform       generate
 application  Application                pkcs11
 protect      Protected by               softcard
 softcard     Soft card to protect key   HashiCorpSC
 recovery     Key recovery               yes
 verify       Verify security of key     yes
 type         Key type                   AES
 size         Key size                   256
 plainname    Key name                   vault_v1_sc
 nvram        Blob in NVRAM (needs ACS)  no
Please enter the pass phrase for softcard `HashiCorpSC':

Please wait........

Key successfully generated.
Path to key: /opt/nfast/kmdata/local/key_pkcs11_uc702bec9b07c5a289ece6aaff880931b5003c8acd-
211aba97b6b5cae5660a9150398566b7ab3e1737
```

5. Create the HMAC key `vault_hmac_v1_sc` using Softcard protection:

```
# generatekey --generate --batch -m1 pkcs11 protect=softcard softcard=HashiCorpSC plainname=vault_hmac_v1_sc
type=HMACSHA256 size=256
key generation parameters:
 operation    Operation to perform       generate
 application  Application                pkcs11
 protect      Protected by               softcard
 softcard     Soft card to protect key   HashiCorpSC
 recovery     Key recovery               yes
 verify       Verify security of key     yes
 type         Key type                   HMACSHA256
 size         Key size                   256
 plainname    Key name                   vault_hmac_v1_sc
 nvram        Blob in NVRAM (needs ACS)  no
Please enter the pass phrase for softcard `HashiCorpSC':

Please wait........

Key successfully generated.
Path to key: /opt/nfast/kmdata/local/key_pkcs11_uc702bec9b07c5a289ece6aaff880931b5003c8acd-
205d6a0baa3957c431fd01986e2ce5046585d3a0
```

6. Create an encryption key `vault_v1_m` and HMAC key `vault_hmac_v1_m` using Module protection:

```
# generatekey --generate --batch -m1 pkcs11 protect=module plainname=vault_v1_m type=AES size=256
key generation parameters:
 operation    Operation to perform       generate
 application  Application                pkcs11
 protect      Protected by               module
 verify       Verify security of key     yes
 type         Key type                   AES
 size         Key size                   256
 plainname    Key name                   vault_v1_m
 nvram        Blob in NVRAM (needs ACS)  no

Key successfully generated.
Path to key: /opt/nfast/kmdata/local/key_pkcs11_ua5274cfcddee6d4a8c2e5226b4e7573633d107fd6
```

```
# generatekey --generate --batch -m1 pkcs11 protect=module plainname=vault_hmac_v1_m type=HMACSHA256 size=256
key generation parameters:
 operation    Operation to perform       generate
 application  Application                pkcs11
 protect      Protected by               module
 verify       Verify security of key     yes
 type         Key type                   HMACSHA256
 size         Key size                   256
 plainname    Key name                   vault_hmac_v1_m
 nvram        Blob in NVRAM (needs ACS)  no

Key successfully generated.
Path to key: /opt/nfast/kmdata/local/key_pkcs11_ua147b4dff28652d4f26de55b7e6ab71ade1e5420f
```

7. Verify the keys created using the `rocs` utility:

```
# rocs
`rocs' key recovery tool
Useful commands: `help', `help intro', `quit'.
rocs> list keys
  No. Name                    App        Protected by
    1 vault_v1_ocs            pkcs11     HashiCorpOCS
    2 vault_hmac_v1_ocs       pkcs11     HashiCorpOCS
    3 vault_v1_sc             pkcs11     HashiCorpSC (HashiCorpSC)
    4 vault_hmac_v1_sc        pkcs11     HashiCorpSC (HashiCorpSC)
    5 vault_v1_m              pkcs11     module
    6 vault_hmac_v1_m         pkcs11     module
rocs> exit
```

8. Verify the keys created using the `nfkminfo` utility.

```
# nfkminfo -l

Keys with module protection:
 key_pkcs11_ua147b4dff28652d4f26de55b7e6ab71ade1e5420f `vault_hmac_v1_m'
 key_pkcs11_ua5274cfcddee6d4a8c2e5226b4e7573633d107fd6 `vault_v1_m'

Keys protected by softcards:
 key_pkcs11_uc702bec9b07c5a289ece6aaff880931b5003c8acd-205d6a0baa3957c431fd01986e2ce5046585d3a0 `vault_hmac_v1_sc'
 key_pkcs11_uc702bec9b07c5a289ece6aaff880931b5003c8acd-211aba97b6b5cae5660a9150398566b7ab3e1737 `vault_v1_sc'

Keys protected by cardsets:
 key_pkcs11_uc5ee369387f1bfa077f9186faaea25b50f1b668dc-5f2a530b5da1ba75861446871e14860e37faf3d3 `vault_v1_ocs'
 key_pkcs11_uc5ee369387f1bfa077f9186faaea25b50f1b668dc-fac560f20f3a5d34ccfd3d288427f57cf3f24cd4 `vault_hmac_v1_ocs'
```

2.5. Verify the PKCS#11 library is available

In the example below an Operator Card Set has been created with the name **HashiCorpOCS**. The card is present in the physical slot (card reader) of the HSM and is loaded to slot **#1**.

1. Execute the `ckcheckinst` command to test the library:

```
# ckcheckinst
PKCS#11 library interface version 2.40
                           flags 0
                    manufacturerID "nCipher Corp. Ltd               "
                libraryDescription "nCipher PKCS#11 13.2.2-134-ab839"
           implementation version 13.02
         Loadsharing and Failover enabled

Slot  Status            Label
====  ======            =====
   0  Fixed token       "loadshared accelerator          "
   1  Operator card     "HashiCorpOCS                    "
   2  Soft token        "HashiCorpSC                     "

Select slot number to run library test or 'R'etry or to 'E'xit:
```

2. Select the slot number of the Operator card and press the **Enter** key:

```
Select slot number to run library test or 'R'etry or to 'E'xit: 2
Using slot number 2.
```

3. Enter the passphrase for the OCS:

```
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.
```

2.6. Find the slot value for each protection method

Each protection method is loaded to a virtual slot. The decimal value of this slot will be needed further down to configure the Vault.

1. Run the `cklist` command. Notice the lines below.

```
# cklist
Listing contents of slot 0
 (token label "loadshared accelerator          ")
...
Listing contents of slot 1
 (token label "HashiCorpOCS                    ")
...
Listing contents of slot 2
 (token label "HashiCorpSC                     ")
```

`loadshared accelerator`

Module protection, that is slot 0.

`HashiCorpOCS`

The name given to the OCS created above, slot 1.

`HashiCorpSC`

The name given to the Softcard token created above, slot 2.

2. Search file `/opt/nfast/log/pkcs11.log` for **pSlotList**. Notice the hex value for each slot. For example:

```
...
2022-06-15 09:21:26 [15724]: pkcs11: 00000000 <    pSlotList[0] 0x2D622495
2022-06-15 09:21:26 [15724]: pkcs11: 00000000 <    pSlotList[1] 0x2D622496
2022-06-15 09:21:26 [15724]: pkcs11: 00000000 <    pSlotList[2] 0x2D622497
...
```

3. Convert to decimal:

Protection Method	Slot Number	Value (Hex)	Value (Decimal)
Module	0	0x2D622495	761406613
OCS	1	0x2D622496	761406614
Softcards	2	0x2D622497	761406615

Note or save the decimal values.

Adding or deleting Softcard tokens, or adding or deleting OCS, or adding or deleting Modules keys will change the values above. Redo the step to find the new values if necessary.

2.7. Create Vault user and group

1. Create the Vault group:

```
# sudo groupadd --system vault
```

2. Create the Vault user:

```
# sudo useradd --system --shell /sbin/nologin --gid vault vault
```

3. Add the Vault user to the nShield **nfast** group:

```
# sudo usermod --append --groups nfast vault
```

2.8. Install Vault

1. Download the Vault package from HashiCorp at https://releases.hashicorp.com/vault/, ensuring that it is the binary file for Enterprise with HSM support:

```
# wget https://releases.hashicorp.com/vault/1.10.0+ent.hsm/vault_1.10.0+ent.hsm_linux_amd64.zip
--2022-04-14 10:39:33-- https://releases.hashicorp.com/vault/1.10.0+ent.hsm/vault_1.10.0+ent.hsm_linux_amd64.zip

Resolving releases.hashicorp.com (releases.hashicorp.com)... 151.101.129.183, 151.101.193.183, 151.101.1.183, ...
Connecting to releases.hashicorp.com (releases.hashicorp.com)|151.101.129.183|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 71617667 (68M) [application/zip]
Saving to: 'vault_1.10.0+ent.hsm_linux_amd64.zip.1'
...
```

2. Unzip the binary file and extract it to the working directory on the host machine, for example `/usr/local/bin`. There should only be a single binary file named `vault`.

```
#  unzip vault_1.10.0+ent.hsm_linux_amd64.zip -d /usr/local/bin
Archive:  vault_1.10.0+ent.hsm_linux_amd64.zip
  inflating: /usr/local/bin/vault
  inflating: /usr/local/bin/TermsOfEvaluation.txt
  inflating: /usr/local/bin/EULA.txt
```

3. Set Vault permissions:

```
# sudo chmod 755 /usr/local/bin/vault
# sudo setcap cap_ipc_lock=+ep /usr/local/bin/vault
# ls -la /usr/local/bin/vault
-rwxr-xr-x. 1 root root 205670424 Mar 21 19:41 /usr/local/bin/vault
```

4. Add the Vault binary file to the path:

```
# sudo vi /etc/profile.d/vault.sh
```

   Add the following information to `vault.sh` and save it:

```
# HashiCorp Vault Enterprise path variable
export PATH="$PATH:/usr/local/bin"
export VAULT_ADDR=http://127.0.0.1:8200
```

5. Create the Vault data directories:

```
# sudo mkdir --parents /opt/vault/data
# sudo mkdir --parents /opt/vault/logs
# sudo chmod --recursive 750 /opt/vault
# sudo chown --recursive vault:vault /opt/vault
```

6. Restart the server.
7. Confirm that the binary file is available:

```
# vault version
Vault v1.10.0+ent.hsm (d71d7710888891761ce43ec4e5f9d9fdeff31d8e) (cgo)
```

2.9. Install the Vault license

1. Open a new terminal and create a directory for the Vault license and configuration files:

```
# sudo mkdir /etc/vault
```

2. Three options are given in the Install a HashiCorp Enterprise License page of the online documentation for enabling an enterprise license, as well as a procedure to request a trail license. For this guide, create a file containing the enterprise license key:

```
# cat /etc/vault/license.hclic
02MV4UU43B...
```

2.10. Create a configuration file

Set up a `/etc/vault/config.hcl` configuration file to enable Vault to be run as a service. See also *Vault commands*.

An example configuration file for using Vault with OCS protection is shown below. The **pin** is the **passphrase** entered when the OCS was created.

```
# PKCS#11 Seal, Entrust nShield Integration
seal "pkcs11" {
lib = "/opt/nfast/toolkits/pkcs11/libcknfast.so"
slot = "761406614"
pin = "hashicorp"
key_label = "vault_v1_ocs"
hmac_key_label = "vault_hmac_v1_ocs"
# Vault is commanding HSM to generate keys if these don't already exists
generate_key = false
}

# Vault listener with TLS disabled
listener "tcp" {
address = "0.0.0.0:8200"
tls_disable = true
}

# Storage
storage "file" {
path = "/opt/vault/data/hsm"
}

ui = true

# License file
license_path = "/etc/vault/license.hclic"

disable_mlock = true
api_addr = "http://127.0.0.1:8200"
cluster_addr = "https://127.0.0.1:8201"

# Managed Key Library
kms_library "pkcs11" {
name = "hsm1" # This can be re-named to anything you like
library = "/opt/nfast/toolkits/pkcs11/libcknfast.so" #PKCS11 Library Location
}
```

In this example:

- The **slot** and **pin** parameters will change according to the protection selected. See section Find the slot value for each protection method.
- The entropy seal mode is set to augmentation. This leverages the HSM for augmenting system entropy via the PKCS#11 protocol.
- The seal wrap is enabled. By enabling seal wrap, Vault wraps your secrets with an extra layer of encryption leveraging the HSM encryption and decryption.
- Notice the path to the license file.

2.11. Create and configure Vault directories

1. Create a vault file in `sysconfig`:

```
# sudo touch /etc/sysconfig/vault
```

2. Create a service file:

```
# vi /etc/systemd/system/vault.service
```

3. Add the following information to the file:

   If deploying on a server with more than two CPUs, you may increase the value of `Environment=GOMAXPROCS` accordingly.

```
[Unit]
Description="HashiCorp Vault"
Requires=network-online.target
After=network-online.target nc_hardserver.service
ConditionFileNotEmpty=/etc/vault/config.hcl
[Service]
User=vault
Group=vault
EnvironmentFile=/etc/sysconfig/vault
ExecStart=/usr/local/bin/vault server -config=/etc/vault/config.hcl
StandardOutput=/opt/vault/logs/output.log
StandardError=/opt/vault/logs/error.log
ExecReload=/bin/kill --signal -HUP $MAINPID
KillMode=process
Restart=on-failure
RestartSec=5
TimeoutStopSec=30
StartLimitInterval=60
StartLimitBurst=3
AmbientCapabilities=CAP_IPC_LOCK
LimitNOFILE=65536
LimitMEMLOCK=infinity
[Install]
WantedBy=multi-user.target
```

4. If you are setting paths different from the default, you must edit the following lines as well in the configuration file:

```
ConditionFileNotEmpty=/etc/vault/config.hcl
EnvironmentFile=-/etc/sysconfig/vault
ExecStart=/opt/vault/bin/vault server -config=/etc/vault/config.hcl
StandardOutput=/opt/vault/logs/output.log
StandardError=/opt/vault/logs/error.log
```

2.12. Enable Vault

1. Set the following environment variable to allow Vault to be accessed from a web browser via the web user interface (web UI). Append the following line to the `/etc/profile.d/vault.sh` file created above, and restart the system:

```
export VAULT_ADDR=http://127.0.0.1:8200
```

2. Enable Vault:

```
# systemctl enable vault.service
Created symlink /etc/systemd/system/multi-user.target.wants/vault.service → /etc/systemd/system/vault.service.
```

2.13. Start Vault

The HSM will be accessed as part of starting Vault. Therefore, the OCS or Softcard is needed.

1. Start the Vault in a separate window.

   If the protection method defined in `/etc/vault/config.hcl` is OCS protection, the OCS card created in the Generate the keys using an OCS protection section must be inserted in the HSM slot, otherwise the Vault will fail to start. The OCS card is not required for the Vault to start if the protection method is Softcard on Module.

```
#  vault server -config=/etc/vault/config.hcl
==> Vault server configuration:

             Api Address: http://127.0.0.1:8200
                     Cgo: enabled
         Cluster Address: https://127.0.0.1:8201
              Go Version: go1.17.7
              Listener 1: tcp (addr: "0.0.0.0:8200", cluster address: "0.0.0.0:8201", max_request_duration: "1m30s",
max_request_size: "33554432", tls: "disabled")
               Log Level: info
                   Mlock: supported: true, enabled: false
           Recovery Mode: false
                 Storage: file
                 Version: Vault v1.10.0+ent.hsm
             Version Sha: d71d7710888891761ce43ec4e5f9d9fdeff31d8e

==> Vault server started! Log data will stream in below:
```

2. Initialize the Vault back in the original window.

   The `vault operator init` command returns the Recovery Key(s) and Initial Root Token. Keep a note of these.

```
# vault operator init
Recovery Key 1: 7YF5RNMkb/KDp+fIS/npxbe+bf3IVcKjRa2Y7jRwZKkm
Recovery Key 2: ugLyy9K+YJyIM/DoBAGLtiN2AD2hwvcI6Sdo9JJgVUT0
Recovery Key 3: oXJ4abPtO+h24DfqPTTS6DUoJZ9Iqome4ztcgr5oVZOP
Recovery Key 4: BJreM1MS4CzjOGui5LDSwXie4X7jLdYNYN4HctXmrIIL
Recovery Key 5: XY9gVqvCObfXVcnF8Wk8BzCi5ppZhDJwUNDbgAF5JMC1

Initial Root Token: hvs.whb6THXfkAfWe4lVXVvVvEuc

Success! Vault is initialized

Recovery key initialized with 5 key shares and a key threshold of 3. Please
securely distribute the key shares printed above.
```

2.14. Log in from the command line

Log in to Vault using the Initial Root Token saved above and save the token below.

```
# vault login hvs.whb6THXfkAfWe4lVXVvVvEuc
Success! You are now authenticated. The token information displayed below
is already stored in the token helper. You do NOT need to run "vault login"
again. Future Vault requests will automatically use this token.

Key                  Value
---                  -----
token                hvs.whb6THXfkAfWe4lVXVvVvEuc
token_accessor       EkvANa3guo6J0HkcxBQ6E82J
token_duration       ∞
token_renewable      false
token_policies       ["root"]
identity_policies    []
policies             ["root"]
```

2.15. Create Managed Key In Vault

1. Create a RSA managed key **hsm-key-ocs-rsa** in Vault that is referencing the key labeled **VaultKeyOCS** created in the nShield HSM. The key **VaultKeyOCS** is protected by the OCS **HashiCorpOCS** in the nShield HSM.

   ```
   # vault write /sys/managed-keys/pkcs11/hsm-key-ocs-rsa library=hsm1 slot=761406614 pin=hashicorp
   key_label="VaultKeyOCSRSA" allow_generate_key=true allow_store_key=true mechanism=0x0001 key_bits=2048
   Success! Data written to: sys/managed-keys/pkcs11/hsm-key-ocs-rsa
   ```

2. Write to the nShield HSM the new managed key **hsm-key-ocs-rsa**.

   ```
   # vault write -f /sys/managed-keys/pkcs11/hsm-key-ocs-rsa/test/sign
   Success! Data written to: sys/managed-keys/pkcs11/hsm-key-ocs-rsa/test/sign
   ```

3. Create a ECDSA managed key **hsm-key-ocs-ecdsa** in Vault that is referencing the key labeled **VaultKeyOCS** created in the nShield HSM.

   ```
   # vault write /sys/managed-keys/pkcs11/hsm-key-ocs-ecdsa library=hsm1 slot=761406614 pin=hashicorp
   key_label="VaultKeyOCSECDSA" allow_generate_key=true allow_store_key=true mechanism=0x1041 curve=P256
   Success! Data written to: sys/managed-keys/pkcs11/hsm-key-ocs-ecdsa
   ```

4. Write to the nShield HSM the new managed key **hsm-key-ocs-ecdsa**.

   ```
   # vault write -f /sys/managed-keys/pkcs11/hsm-key-ocs-ecdsa/test/sign
   Success! Data written to: sys/managed-keys/pkcs11/hsm-key-ocs-ecdsa/test/sign
   ```

5. List all keys created in the nShield HSM. Notice the new keys **VaultKeyOCSRSA** and **VaultKeyOCSECDSA**

```
# nfkminfo -l
...
Keys protected by cardsets:
 key_pkcs11_uc5ee369387f1bfa077f9186faaea25b50f1b668dc-3a2847fc5c927a0b3cb223eeb21eaed86e8dfc2b `VaultKeyOCSECDSA'
 key_pkcs11_uc5ee369387f1bfa077f9186faaea25b50f1b668dc-5f2a530b5da1ba75861446871e14860e37faf3d3 `vault_v1_ocs'
 key_pkcs11_uc5ee369387f1bfa077f9186faaea25b50f1b668dc-f456a657cc0ccb300cd51c5d27eb180d1f3890db `VaultKeyOCSRSA'
 key_pkcs11_uc5ee369387f1bfa077f9186faaea25b50f1b668dc-fac560f20f3a5d34ccfd3d288427f57cf3f24cd4 `vault_hmac_v1_ocs'
```

6. Enable the PKI secrets engine at the path `pki` and reference a managed key **hsm-key** stored in the HSM.

```
# vault secrets enable -path=pki -allowed-managed-keys=hsm-key pki
Success! Enabled the pki secrets engine at: pki/
```

7. Perform PKI operations as needed. See the PKI Secrets Engine page in the online documentation.

Appendix A: Troubleshooting

Error Message	Resolution
Vault fails to start. There may not be a log file created if the vault fails to start upon executing `# systemctl start vault.service`.	Execute the following instead to get some debugging information. `# vault server -config=/etc/vault/config.hcl`.
Error: **failed to decrypt encrypted stored keys: error initializing session for decryption: error logging in to HSM: pkcs11: 0xE0: CKR_TOKEN_NOT_PRESENT**	Ensure that the Operator card is inserted in the physical slot of the nShield HSM.

Appendix B: Vault commands

B.1. Vault commands

Task	Command
Log into Vault	`# vault login s.InitialRootToken`
Check Vault status	`# vault status`
Unseal Vault	`# vault operator unseal -address=http://127.0.0.1:8200`
Seal Vault	`# vault operator seal`

B.2. vault.service commands

Task	Command
Enable Vault Service	`# systemctl enable vault.service`
Disable Vault service	`# systemctl disable vault.service`
Start Vault service	`# systemclt start vault.service`
Stop Vault service	`# systemclt stop vault.service`
Restart Vault service	`# systemctl restart vault.service`