# Contents

1. Introduction .................................................................................................................. 3  
   1.1. Product configurations ......................................................................................... 3  
   1.2. Requirements ....................................................................................................... 4  
   1.3. More information .................................................................................................. 5  
2. Integration procedures .................................................................................................. 6  
   2.1. System preparation ............................................................................................... 6  
   2.2. Install the HSM .................................................................................................... 7  
   2.3. Install the Security World software and create a Security World ....................... 7  
   2.4. Generate the Vault encryption and HMAC keys with OCS and Softcard protection ......................................................................................................................... 8  
   2.5. Verify the PKCS#11 library is available ............................................................... 13  
   2.6. Find the slot value for each protection method .................................................. 14  
   2.7. Create Vault user and group ............................................................................... 15  
   2.8. Install Vault ......................................................................................................... 15  
   2.9. Install the Vault license ....................................................................................... 17  
   2.10. Create a configuration file .................................................................................. 17  
   2.11. Create and configure Vault directories ............................................................. 18  
   2.12. Enable Vault ...................................................................................................... 19  
   2.13. Start Vault ......................................................................................................... 20  
   2.14. Log in from the command line ........................................................................... 21  
   2.15. Create Managed Key In Vault .......................................................................... 21  
3. Appendix A: Troubleshooting ......................................................................................... 23  
4. Appendix B: Vault commands ....................................................................................... 24  
   B.1. Vault commands ................................................................................................. 24  
   B.2. vault.service commands ..................................................................................... 24
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:

<table>
<thead>
<tr>
<th>Product</th>
<th>Version</th>
</tr>
</thead>
<tbody>
<tr>
<td>HashiCorp Vault Enterprise</td>
<td>v1.10.0 Enterprise HSM</td>
</tr>
<tr>
<td>Base OS</td>
<td>Red Hat Enterprise 8.4</td>
</tr>
</tbody>
</table>

1.1.1. Supported nShield features

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

<table>
<thead>
<tr>
<th>Feature</th>
<th>Support</th>
</tr>
</thead>
<tbody>
<tr>
<td>Softcards</td>
<td>Yes</td>
</tr>
<tr>
<td>Module Only Key</td>
<td>Yes</td>
</tr>
<tr>
<td>OCS cards</td>
<td>Yes</td>
</tr>
<tr>
<td>nSaaS</td>
<td>Supported but not tested</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Security World Software</th>
<th>Firmware</th>
<th>Netimage</th>
<th>OCS</th>
<th>Softcard</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>13.2.2 (FIPS Pending)</td>
<td>13.2.2</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

1.1.2.2. Connect XC

<table>
<thead>
<tr>
<th>Security World Software</th>
<th>Firmware</th>
<th>Netimage</th>
<th>OCS</th>
<th>Softcard</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>12.50.11 (FIPS Certified)</td>
<td>12.80.4</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>12.60.15 (CC Certified)</td>
<td>12.80.4</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

1.1.2.3. Connect +

<table>
<thead>
<tr>
<th>Security World Software</th>
<th>Firmware</th>
<th>Netimage</th>
<th>OCS</th>
<th>Softcard</th>
<th>Module</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>12.50.8 (FIPS Certified)</td>
<td>12.80.4</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td></td>
<td>2.55.4 (CC Certified)</td>
<td>12.45.1</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>

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 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 information about limitations on FIPS authorization, see the Installation Guide of the nShield HSM.

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

• 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:

- How to locally set up a new or replacement nShield Connect
- How to remotely set up a new or replacement nShield Connect
- How to remotely set up a new or replacement nShield Connect XC Serial Console model

Access to the Entrust nShield Support Portal is available to customers under maintenance. To request an account, contact nshield.support@entrust.com.

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:

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

   Add the following info to nfast.sh and save:

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

4. Restart the server.

5. Confirm that the HSM is available:

   ```bash
   # 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 2f9718083b4c0d9d4f30913d0a340fffe71aa576
hkmm 9db75036f724f02378d3b2e643b9af3f38369a (type Rijndael)
hkmuk c2be99fe177f1b754d48e2f2d2df0f8c9692cb
hkres 78a56646d655bfe89477ba85448b3fbdabaa87
hkmra a3cc2e287a0d63b3b288d2b9049d4768a4d34b
hkfps 5ab63d1f858c3d5485ee1d6f1f8ca74a74a390
hkmrc ec1385f4e37bba2f00ba8093b5b780ede4d30048
hkrtc 3942ce6716527f2e96708ed281b363f599ecc
hknv 5c63f4d18ecdc206b3c4d31eced229560958c598
hknv f48579562116cd04d8544708599c6f98ad1db
hknnull 0000000000000000000000000000000000000000
ex.client none
k-out-of-n 1/2
other quota m=1 r=1 nv=1 rtc=1 dssee=1 fto=1
createtime 2022-02-04 19:08:39
nsos 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 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 = 5ee369387f1bfa077f9186f0aae25b50f1b668dc
   ```

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
   5ee369387f1bfa077f9186f0aae25b50f1b668dc  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

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_4cf560f20f3a5d34ccfd3d288427f57cf3f24cd4

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_4cf560f20f3a5d34ccfd3d288427f57cf3f24cd4

2.4.2. Generate the keys using Softcard and Module protection

1. Create a `/opt/nfast/cknfastrc` file with the following content:
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 702bec9b07c5a289ece6aafff880931b5003c8acd
```

3. List the Softcard created.

```
# nfkminfo -s
SoftCard summary - 1 softcards:
Operator logical token hash               name
702bec9b07c5a289ece6aafff880931b5003c8acd  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_uc702bec9b07c5a289ece6aafff880931b5003c8acd-211aba9b5caee5660a915a939e56d97bab3e1737
```

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
vram        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_uc702bec9b07c5a289ece6aaff88031b5003c8acd-205d6aa3957c431fd81986e2e5b465585d3a0

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
vram        Blob in NVRAM (needs ACS)  no

Key successfully generated.
Path to key: /opt/nfast/kmdata/local/key_pkcs11_ua5274cfcedee64d4a8c2e5226b4e7573633d107fd6

# 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
vram        Blob in NVRAM (needs ACS)  no

Key successfully generated.
Path to key: /opt/nfast/kmdata/local/key_pkcs11_ua147b4dfff286526f26de55b765a71ade1e5420f

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

```
# nfkminfo -l
Keys with module protection:
key_pkcs11_ua147b4df28b52df26de55b7e6ab71ade1e5420f 'vault_hmac_v1_m'
key_pkcs11_ua5274cf7e26e4757363d107fd6 'vault_v1_m'

Keys protected by softcards:
key_pkcs11_uc702bec9b07c5a289e6aaa9574314d01986e2ce504585d3a0 'vault_hmac_v1_sc'
key_pkcs11_uc702bec9b07c5a289e6aaa9574314d01986e2ce504585d3a0 'vault_v1_sc'

Keys protected by cardsets:
key_pkcs11_uc5ee369387f1bfa877f9186faa25b8f1b668dc-5f2a530b5d5a1ba75861446871e1486e37f3f3d 'vault_v1_ocs'
key_pkcs11_uc5ee369387f1bfa877f9186faa25b8f1b668dc-5f2a530b5d5a1ba75861446871e1486e37f3f3d '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:

<table>
<thead>
<tr>
<th>Test</th>
<th>Pass/Failed</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Generate RSA key pair</td>
<td>Pass</td>
</tr>
<tr>
<td>2 Generate DSA key pair</td>
<td>Pass</td>
</tr>
<tr>
<td>3 Encryption/Decryption</td>
<td>Pass</td>
</tr>
<tr>
<td>4 Signing/Verification</td>
<td>Pass</td>
</tr>
<tr>
<td>Deleting test keys</td>
<td>ok</td>
</tr>
</tbody>
</table>

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.

   For example:
3. Convert to decimal:

<table>
<thead>
<tr>
<th>Protection Method</th>
<th>Slot Number</th>
<th>Value (Hex)</th>
<th>Value (Decimal)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Module</td>
<td>0</td>
<td>0x2D622495</td>
<td>761406613</td>
</tr>
<tr>
<td>OCS</td>
<td>1</td>
<td>0x2D622496</td>
<td>761406614</td>
</tr>
<tr>
<td>Softcards</td>
<td>2</td>
<td>0x2D622497</td>
<td>761406615</td>
</tr>
</tbody>
</table>

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
   ```

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:
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`.

3. Set Vault permissions:

4. Add the Vault binary file to the path:

5. Create the Vault data directories:

6. Restart the server.

7. Confirm that the binary file is available:
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.helic
   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.
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`:

   ```bash
   # 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:
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: d71d7710888891761ce43ec4e5f9d9fdeff31d0e
   ===> 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: 7YF5RMNkbkX0dp+flS/spxbe+bF31IVcKjRa2Y7jrZmZkm
   Recovery Key 2: ugLy9hK+YjyIM/DozAgLtiNZAg2hvvc6Sdo933qVUT0
   Recovery Key 3: oxJ4abPrt0+h2408qTT56DU0jZ91qone4ztcqvoZP
   Recovery Key 4: B1r8MGl5Ac7jO6ih5L3WxXie4k7jLdYNYVH4HctxMrIIL
   Recovery Key 5: Xy9gVqCQbfXCVcnF8Wk8BzC1p3p2hDjUNDbgAF5JNC1
   Initial Root Token: hvs.w6e6THXkAFMe41VXvVVeuc
   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.whb6THXfkAfWx4lVXVvVvEuc
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.
```

<table>
<thead>
<tr>
<th>Key</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>---</td>
<td>-----</td>
</tr>
<tr>
<td>token</td>
<td>hvs.whb6THXfkAfWx4lVXVvVvEuc</td>
</tr>
<tr>
<td>token_accessor</td>
<td>EkvAa3guo6J0HkcxBQ5E82J</td>
</tr>
<tr>
<td>token_duration</td>
<td>∞</td>
</tr>
<tr>
<td>token_renewable</td>
<td>false</td>
</tr>
<tr>
<td>token_policies</td>
<td>[&quot;root&quot;]</td>
</tr>
<tr>
<td>identity_policies</td>
<td>[]</td>
</tr>
<tr>
<td>policies</td>
<td>[&quot;root&quot;]</td>
</tr>
</tbody>
</table>

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`
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
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

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

B.1. Vault commands

<table>
<thead>
<tr>
<th>Task</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Log into Vault</td>
<td><code># vault login s.InitialRootToken</code></td>
</tr>
<tr>
<td>Check Vault status</td>
<td><code># vault status</code></td>
</tr>
<tr>
<td>Unseal Vault</td>
<td><code># vault operator unseal -address=http://127.0.0.1:8200</code></td>
</tr>
<tr>
<td>Seal Vault</td>
<td><code># vault operator seal</code></td>
</tr>
</tbody>
</table>

B.2. vault.service commands

<table>
<thead>
<tr>
<th>Task</th>
<th>Command</th>
</tr>
</thead>
<tbody>
<tr>
<td>Enable Vault Service</td>
<td><code># systemctl enable vault.service</code></td>
</tr>
<tr>
<td>Disable Vault service</td>
<td><code># systemctl disable vault.service</code></td>
</tr>
<tr>
<td>Start Vault service</td>
<td><code># systemctl start vault.service</code></td>
</tr>
<tr>
<td>Stop Vault service</td>
<td><code># systemctl stop vault.service</code></td>
</tr>
<tr>
<td>Restart Vault service</td>
<td><code># systemctl restart vault.service</code></td>
</tr>
</tbody>
</table>