THALES

HashiCorp Vault: Integration Guide

THALES LUNA HSM & DPOD LUNA CLOUD HSM

Document Information

Document Part Number	007-000264-001
Revision	D
Release Date	3 December 2020

Trademarks, Copyrights, and Third-Party Software

CONTENTS

Overview

This document describes how to store the HashiCorp Vault encryption key on a Thales Luna HSM or Luna Cloud HSM service and to leverage HSM for entropy augmentation. HashiCorp Vault Enterprise allows HSM support as a feature. It uses the HSM for:

- **Master Key Wrapping**: HashiCorp Vault protects its master key by transiting it through the HSM for encryption rather than splitting into key shares.
- **Automatic Unsealing**: HashiCorp Vault stores its HSM-wrapped master key in storage, allowing for automatic unsealing.
- **Seal Wrapping**: Provides FIPS key storage conforming functionality for critical security parameters.
- **Entropy Augmentation:** HashiCorp Vault leverages HSM for augmenting system entropy via the PKCS#11 protocol.

The benefits of securing the keys with Luna HSMs include:

- Secure generation, storage and protection of the encryption keys on FIPS 140-2 level 3 validated hardware.
- Full life cycle management of the keys.
- Access to the HSM audit trail*.
- Take advantage of cloud services with confidence.

*Luna Cloud HSM service does not have access to the secure audit trail

Certified Platforms

This integration is certified on the following platforms.

HSM Type	Platforms Tested
Luna HSM, Luna Cloud HSM	RHEL

Luna HSM: Luna HSM appliances are purposefully designed to provide a balance of security, high performance, and usability that makes them an ideal choice for enterprise, financial, and government organizations. Luna HSMs physically and logically secure cryptographic keys and accelerate cryptographic processing. The Luna HSM on premise offerings include the Luna Network HSM, Luna PCIe HSM, and Luna USB HSMs. Luna HSMs are also available for access as an offering from cloud service providers such as IBM cloud HSM and AWS cloud HSM classic

Luna Cloud HSM: Luna Cloud HSM platform provides on-demand, cloud-based HSM and Key Management services through a simple graphical user interface. With Luna Cloud HSM, security is simple, cost effective and easy to manage because there is no hardware to buy, deploy and maintain. As an Application Owner, you click and deploy services, generate usage reports and maintain just the services you need.

Prerequisites

Before you proceed with the integration, complete the following tasks:

Configure Luna HSM

If you are using Luna HSM, complete the following:

1. Verify the HSM is set up, initialized, provisioned and ready for deployment. Refer to the *Luna HSM Product Documentation* for more information.
2. Create a partition on the HSM that will be later used by HashiCorp Vault.
3. If using a Luna Network HSM, register a client for the system and assign the client to the partition to create an NTLS connection. Initialize the Crypto Officer and Crypto User roles for the registered partition.
4. Ensure that the partition is successfully registered and configured. The command to see the registered partition is:

```
# /usr/safenet/lunaclient/bin/lunacm

lunacm (64-bit) v10.2.0-111. Copyright (c) 2020 SafeNet. All rights reserved.

Available HSMs:

Slot Id ->              0

Label ->                HashiCorp Vault

Serial Number ->        1280780175943

Model ->                LunaSA 7.4.0

Firmware Version ->     7.4.0

Configuration ->        Luna User Partition With SO (PW) Key Export With
                        Cloning Mode

Slot Description ->     Net Token Slot

FM HW Status ->         FM Ready

Current Slot Id: 0
```

5. For PED-authenticated HSM, enable partition policies 22 and 23 to allow activation and auto-activation.

> **NOTE:** Refer to Luna HSM documentation for detailed steps on creating NTLS connection, initializing the partitions, and assigning various user roles.

Set up Luna HSM High-Availability

Refer to the Luna HSM documentation for HA steps and details regarding configuring and setting up two or more HSM boxes on host systems. You must enable the HAOnly setting in HA for failover to work so that if the primary goes down due to any reason all calls automatically route to the secondary until the primary recovers and starts up.

Set up Luna HSM in FIPS Mode

Under FIPS 186-3/4, the RSA methods permitted for generating keys are 186-3 with primes and 186-3 with aux primes. This means that RSA PKCS and X9.31 key generation is no longer approved for operation in a FIPS-compliant HSM. If you are using the Luna HSM in FIPS mode, you have to make the following change in the configuration file:

For Linux:

```
Misc = {
RSAKeyGenMechRemap = 1;
}
```

```
For Windows:
```

```
[Misc]
RSAKeyGenMechRemap=1
```

The above setting redirects the older calling mechanism to a new approved mechanism when Luna HSM is in FIPS mode.

> **NOTE:** The above setting is not required for Universal Client. This setting is applicable only for Luna Client 7.x.

Configure Luna Cloud HSM Service

You can configure Luna Cloud HSM Service in the following ways:

- Standalone Cloud HSM service using minimum client package
- Standalone Cloud HSM service using full Luna client package
- Luna HSM and Luna Cloud HSM service in hybrid mode

> **NOTE:** Luna Client v10.x or higher is required for configuring Luna HSM device and Luna Cloud HSM service in hybrid mode.

Standalone Cloud HSM service using minimum client package

To configure Luna Cloud HSM service using minimum client package:

1. Transfer the downloaded .zip file to your Client workstation using pscp, scp, or other secure means.
2. Extract the .zip file into a directory on your client workstation.
3. Extract or untar the appropriate client package for your operating system. Do not extract to a new subdirectory; place the files in the client install directory.

   ```
   [Windows]
   cvclient-min.zip
   [Linux]
   cvclient-min.tar
   # tar -xvf cvclient-min.tar
   ```

4. Run the **setenv** script to create a new configuration file containing information required by the Luna Cloud HSM service.

   ```
   [Windows]
   Right-click setenv.cmd and select Run as Administrator.
   [Linux]
   Source the setenv script.
   ```

```
# source ./setenv
```

5. Run the **LunaCM** utility and verify the Cloud HSM service is listed.

Standalone Cloud HSM service using full Luna client package

To configure Luna Cloud HSM service using full Luna client package:

1. Transfer the downloaded .zip file to your Client workstation using pscp, scp, or other secure means.
2. Extract the .zip file into a directory on your client workstation.
3. Extract or untar the appropriate client package for your operating system. Do not extract to a new subdirectory; place the files in the client install directory.

   ```
   [Windows]
   cvclient-min.zip
   [Linux]
   cvclient-min.tar
   # tar -xvf cvclient-min.tar
   ```

4. Run the **setenv** script to create a new configuration file containing information required by the Luna Cloud HSM service.

   ```
   [Windows]
   Right-click setenv.cmd and select Run as Administrator.
   [Linux]
   Source the setenv script.
   # source ./setenv
   ```

5. Copy the server and partition certificates from the Cloud HSM service client directory to Luna client certificates directory:

   Cloud HSM Certificates:

   ```
   server-certificate.pem
   partition-ca-certificate.pem
   partition-certificate.pem
   ```

   LunaClient Certificate Directory:

   ```
   [Windows default location for Luna Client]
   C:\Program Files\Safenet\Lunaclient\cert\
   [Linux default location for Luna Client]
   /usr/safenet/lunaclient/cert/
   ```

   NOTE: Skip this step for Luna Client v10.2 or higher.

6. Open the configuration file from the Cloud HSM service client directory and copy the **XTC** and **REST** section.

   ```
   [Windows]
   crystoki.ini
   ```

```
[Linux]

Chrystoki.conf
```

7. Edit the Luna Client configuration file and add the **XTC** and **REST** sections copied from Cloud HSM service client configuration file.
8. Change server and partition certificates path from step 5 in **XTC** and **REST** sections. Do not change any other entries provided in these sections.

   [XTC]

```
. . .
PartitionCAPath=<LunaClient_cert_directory>\partition-ca-certificate.pem
PartitionCertPath00=<LunaClient_cert_directory>\partition-certificate.pem
. . .

[REST]
. . .
SSLClientSideVerifyFile=<LunaClient_cert_directory>\server-certificate.pem
. . .
```

   NOTE: Skip this step for Luna Client v10.2 or higher.

9. Edit the following entry from the **Misc** section and update the correct path for the **plugins** directory:

```
Misc]
PluginModuleDir=<LunaClient_plugins_directory>

[Windows Default]

C:\Program Files\Safenet\Lunaclient\plugins\

[Linux Default]

/usr/safenet/lunaclient/plugins/
```

   Save the configuration file. If you wish, you can now safely delete the extracted Cloud HSM service client directory.

10. Reset the **ChrystokiConfigurationPath** environment variable and point back to the location of the Luna Client configuration file.

    [Windows]

    In the Control Panel, search for "environment" and select **Edit the system environment variables**. Click **Environment Variables**. In both list boxes for the current user and system variables, edit **ChrystokiConfigurationPath** and point to the **crystoki.ini** file in the Luna client install directory.

    [Linux]

    Either open a new shell session, or export the environment variable for the current session pointing to the location of the **Chrystoki.conf** file:

```
# export ChrystokiConfigurationPath=/etc/
```

11. Run the **LunaCM** utility and verify that the Cloud HSM service is listed. In hybrid mode, both Luna and Cloud HSM service will be listed.

> **NOTE:** Follow the Luna Cloud HSM documentation for detailed steps for creating service, client, and initializing various user roles.

Luna HSM and Luna Cloud HSM service in hybrid mode

To configure Luna HSM and Luna Cloud HSM service in hybrid mode, follow the steps mentioned under the Standalone Cloud HSM service using full Luna client package section above.

> **NOTE:** Luna Client v10.x or higher is required for configuring Luna HSM device and Luna Cloud HSM service in hybrid mode.

Luna Cloud HSM Service in FIPS mode

Luna Cloud HSM service operates in both FIPS and non-FIPS mode. If your organization requires non-FIPS algorithms for your operations, ensure you enable the **Allow non-FIPS approved algorithms** check box when configuring your Cloud HSM service. The FIPS mode is enabled by default. Refer to the Mechanism List in the SDK Reference Guide for more information about available FIPS and non-FIPS algorithms.

Set up HashiCorp Vault

Download and install the HashiCorp Vault and set up the system environment to support the integration. HashiCorp Vault is distributed as a binary package for all supported platforms. The HashiCorp Vault is packaged as a zip archive. For more details, refer to the *HashiCorp Documentation*.

1. Download the HashiCorp Vault package from HashiCorp.
2. Unzip the package in the working directory on the host machine. HashiCorp Vault runs as a single binary named `vault`.
3. Add the current working directory to the PATH so that `vault` is executable from any directory.
4. After installing `vault`, verify the installation worked by opening a new terminal session and checking that the vault binary is available. By executing `vault`, you should see help output similar to the following:

```
Usage: vault <command> [args]

Common commands:
    read        Read data and retrieves secrets
    write       Write data, configuration, and secrets
    delete      Delete secrets and configuration
    list        List data or secrets
    login       Authenticate locally
    agent       Start a Vault agent
    server      Start a Vault server
    status      Print seal and HA status
    unwrap      Unwrap a wrapped secret

Other commands:
    audit           Interact with audit devices
    auth            Interact with auth methods
    kv              Interact with Vault's Key-Value storage
    lease           Interact with leases
    namespace       Interact with namespaces
    operator        Perform operator-specific tasks
    path-help       Retrieve API help for paths
    plugin          Interact with Vault plugins and catalog
    policy          Interact with policies
    secrets         Interact with secrets engines
    ssh             Initiate an SSH session
    token           Interact with tokens
```

Integrating HashiCorp Vault with Luna HSMs

To set up HashiCorp Vault using a Luna HSM or Luna Cloud HSM service, complete the following tasks:

> Enable PKCS11 seal
> Configure Entropy Augmentation
> Start the Vault
> Initialize the Vault
> Log into the Vault
> Use the Secret Engine
> Enable Entropy Augmentation

Enable the PKCS11 seal

The PKCS11 seal configures HashiCorp Vault to use an HSM with PKCS11 as the seal wrapping mechanism. To enable the PKCS11 seal, create the HashiCorp Vault's configuration file named `config.json` and specify the **seal**,**storage** and **listener** stanzas:

```
# PKCS11 seal
seal "pkcs11" {
  lib = "<path to cryptoki library>"
  slot = "<slot number>"
  pin = "<partition password>"
  key_label = "HashiCorp"
  hmac_key_label = "HashiCorp_hmac"
  generate_key = "true"
}

storage "file" {
  path = "/tmp/vault"
}

# Addresses and ports on which Vault will respond to requests
listener "tcp" {
  address     = "127.0.0.1:8200"
  tls_disable = "true"
}
ui = true
```

Where:

`lib`: The path to the PKCS#11 library shared object file.

`slot` : HSM partition slot number.

`pin`: HSM partition password

`generate_key`: It instructs Vault to generate a key if no existing key with the label specified by `key_label` can be found at Vault initialization time.

`hmac_key_label`: The label of the key to use for HMACing

`mechanism`: The encryption/decryption mechanism to use, specified as a decimal or hexadecimal (prefixed by 0x) string.

Currently supported mechanisms (in order of precedence):

```
0x1085 CKM_AES_CBC_PAD (HMAC mechanism required)

0x1082 CKM_AES_CBC (HMAC mechanism required)

0x1087 CKM_AES_GCM

0x0009 CKM_RSA_PKCS_OAEP

0x0001 CKM_RSA_PKCS
```

Below is an example to define the RSA_PKCS_OEAP mechanism for Master Key Wrapping in `config.json` file.

```
# PKCS11 seal
seal "pkcs11" {
  lib = "<path to cryptoki library>"
  slot = "<slot number>"
  pin = "<partition password>"
  key_label = "rsa_oaep_key"
  mechanism = "0x0009"
  rsa_oaep_hash = "sha256"
  generate_key = "true"
}

storage "file" {
  path = "/tmp/vault"
}

# Addresses and ports on which Vault will respond to requests
listener "tcp" {
  address     = "127.0.0.1:8200"
  tls_disable = "true"
}
ui = true
```

NOTE: Alternatively, the HSM seal can be activated by providing the following environment variables:

VAULT_HSM_LIB, VAULT_HSM_SLOT, VAULT_HSM_PIN, VAULT_HSM_KEY_LABEL, VAULT_HSM_HMAC_KEY_LABEL VAULT_HSM_MECHANISM and VAULT_HSM_GENERATE_KEY

Although the configuration file allows you to pass in VAULT_HSM_PIN as part of the seal's parameters, it is strongly recommended to set this value via environment variables.

Configure Entropy Augmentation

Vault Enterprise version 1.3 introduced the Entropy Augmentation function to leverage HSM for augmenting system entropy via the PKCS#11 protocol. To configure Entropy Augmentation for Vault version 1.3 and above, define the **entropy** stanza in server configuration file `config.json`.

NOTE: Since Vault will delegate the random number generation to the HSM, be sure to set the seal stanza with HSM cluster connection information.

Start the Vault

Start the Vault server using the configuration file.

```
# ./vault server -config config.json
```

```
[root@localhost home]# ./vault server -config config.json
==> Vault server configuration:

    HSM PKCS#11 Version: 2.20
            HSM Library: Chrystoki
    HSM Library Version: 10.1
    HSM Manufacturer ID: SafeNet
               HSM Type: pkcs11
                    Cgo: enabled
             Listener 1: tcp (addr: "127.0.0.1:8200", cluster address: "127.0.0.1:8201",
              Log Level: info
                  Mlock: supported: true, enabled: true
          Recovery Mode: false
                Storage: file
                Version: Vault v1.3.0+ent.hsm

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

Initialize the Vault

You must initialize the Vault before you may access it to begin configuring and managing secrets. When Vault is initialized while using an HSM, rather than unseal keys being returned to the operator, recovery keys are returned. Some Vault operations such as generation of a root token require these recovery keys. To initialize the HashiCorp Vault:

1. Launch a new terminal session and execute the following command:

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

2. Check the status of Vault by executing :

   ```
   # ./vault status
   ```

```
Key                    Value
---                    -----
Recovery Seal Type     pkcs11
Initialized            false
Sealed                 true
Total Recovery Shares  0
Threshold              0
Unseal Progress        0/0
Unseal Nonce           n/a
Version                n/a
HA Enabled             false
```

3. Initialize the Vault by executing :

```
# ./vault operator init -recovery-shares=1 -recovery-threshold=1
```

This will generate a recovery key and initial root token. Copy these keys and keep it in safe place.

```
Recovery Key 1: vSz2okfEnhyEqXT4gXTUJDTVo5bLa7qrYSVtZEhi5ZA=

Initial Root Token: 5tvLldtsxhKxVAGenD5abbHI

Success! Vault is initialized

Recovery key initialized with 1 key shares and a key threshold of 1. Please
securely distribute the key shares printed above.
```

Note the following logs in the first terminal where Vault Server is running:

```
2018-11-16T06:15:48.859-0500 [INFO]  core: loaded wrapping token key
2018-11-16T06:15:48.860-0500 [INFO]  core: successfully mounted backend:
type=kv path=secret/
…
2018-11-16T06:15:48.952-0500 [INFO]  core: root token generated
…
2018-11-16T06:15:49.031-0500 [INFO]  core: vault is unsealed
2018-11-16T06:15:49.032-0500 [INFO]  core: post-unseal setup starting
2018-11-16T06:15:49.153-0500 [INFO]  core: loaded wrapping token key
…
2018-11-16T06:15:49.157-0500 [INFO]  core: successfully unsealed with stored
key(s): stored_keys_used=1
2018-11-16T06:15:49.157-0500 [INFO]  expiration: lease restore complete
```

4. Verify the keys generated on the partition by executing `partition contents` in lunacm.

```
lunacm:>partition contents

       The 'Crypto Officer' is currently logged in.  Looking for objects
       accessible to the 'Crypto Officer'.

       Object list:

       Label:         hashicorp_hmac
       Handle:        97
       Object Type:   Symmetric Key
       Object UID:    350000001800000025b990800

       Label:         hashicorp
       Handle:        85
       Object Type:   Symmetric Key
       Object UID:    340000001800000025b990800


       Number of objects:  2
```

Log into the Vault

You must log in to the Vault to begin configuring and managing the secrets engine. To log in to the HashiCorp Vault:

1. Log into the Vault.

   ```
   # ./vault login <VAULT-TOKEN>
   ```

   where `<VAULT-TOKEN>` is the initial root token generated during Vault initialization.

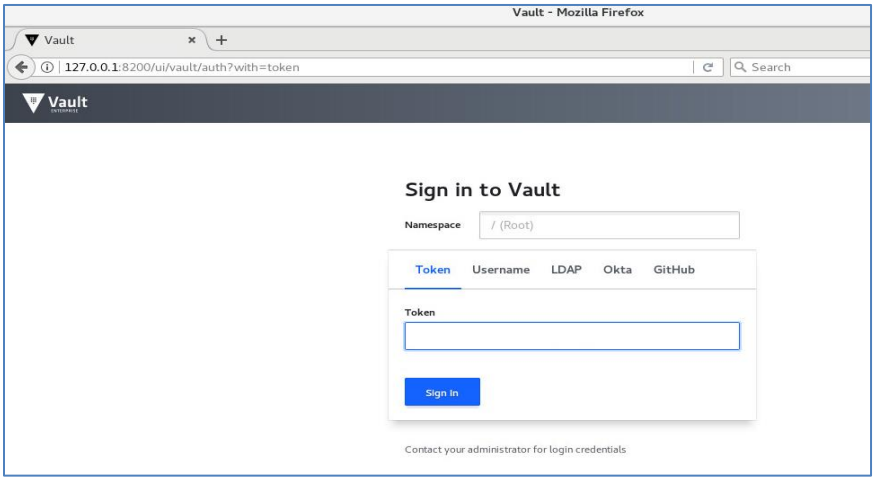
```
[root@hashicorp]# ./vault login 5tvLldtsxhKxVAGenD5abbHI
Success! You are now authenticated. The token information displayed below
is already stored in the token helper. You do NOT need to run "vault login"
again. Future Vault requests will automatically use this token.

Key                  Value
---                  -----
token                5tvLldtsxhKxVAGenD5abbHI
token_accessor       94w73PrwFKjk8b0MyaMmPRay
token_duration       ∞
token_renewable      false
token_policies       ["root"]
identity_policies    []
policies             ["root"]
```

2. Check to verify that the Vault server is now initialized and auto-unsealed.

```
[root@localhost home]#  ./vault status
Key                      Value
---                      -----
Recovery Seal Type       shamir
Initialized              true
Sealed                   false
Total Recovery Shares    1
Threshold                1
Version                  1.3.0+ent.hsm
Cluster Name             vault-cluster-f52d12c3
Cluster ID               3085e6c5-c2b7-fdd0-8871-a826bbf86916
HA Enabled               false
```

3. If UI is enabled in configuration file (`config.json`), the Vault UI can be accessed:

   http://127.0.0.1:8200/ui



4. Provide the VAULT TOKEN in the field of **Token** to login.

Use the Secrets Engine

Secrets engines are components which store, generate, or encrypt data. The `kv` secrets engine is used to store arbitrary secrets within the configured physical storage for Vault. Versioning can be enabled and a configurable number of versions for each key will be stored. To use the Secrets Engine:

1. Execute the following command to view the secrets:

```
# ./vault secrets list
```

```
[root@hashicorp]# ./vault secrets list
Path          Type         Accessor              Description
----          ----         --------              -----------
cubbyhole/    cubbyhole    cubbyhole_16b25559    per-token private secret storage
identity/     identity     identity_a6ee82e3     identity store
secret/       kv           kv_70be3e30           key/value secret storage
sys/          system       system_494262c6       system endpoints used for control, policy and debugging
```

2. Enable the kv engine

```
# ./vault secrets enable -version=1 kv

Success! Enabled the kv secrets engine at: kv/
```

3. Write arbitrary secret data

```
# ./vault kv put kv/my-secret my-value=s3cr3t

Success! Data written to: kv/my-secret
```

4. Display the secret value by executing command:

```
 # ./vault kv get kv/my-secret

====== Data ======
Key          Value
---          -----
my-value     s3cr3t
```

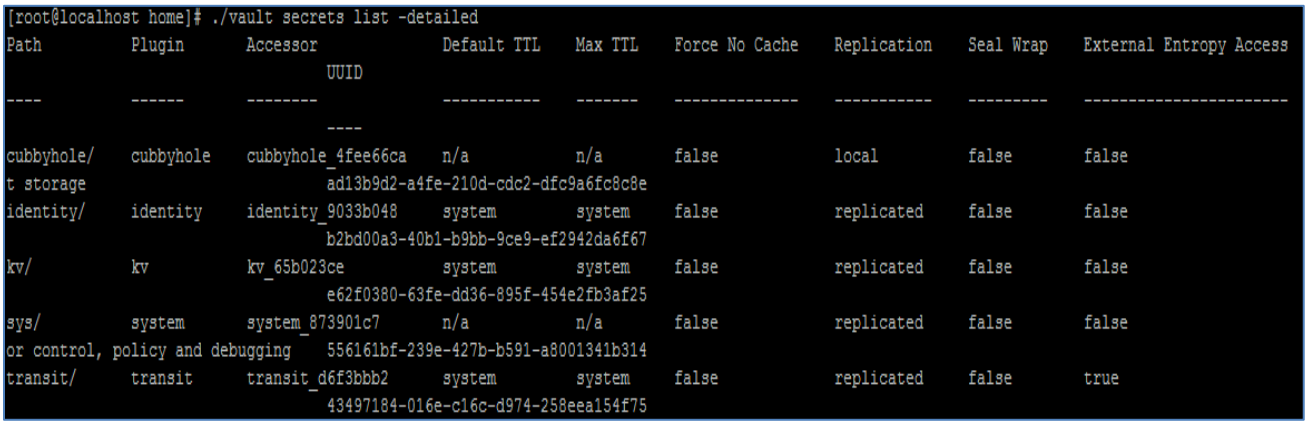
Enable Entropy Augmentation

To leverage the external entropy source, set the `external_entropy_access` parameter to true while enabling a `secrets` engine or `auth` method. To enable external entropy source on a transit secrets engine:

1. Enable `transit` secrets engine with external entropy source.

```
# ./vault secrets enable -external-entropy-access transit
Success! Enabled the transit secrets engine at: transit/
```

2. List the enabled secrets engine with `-detailed` flag.



```
[root@localhost home]# ./vault secrets list -detailed
Path          Plugin       Accessor             Default TTL    Max TTL    Force No Cache    Replication    Seal Wrap    External Entropy Access
                           UUID
----          ------       --------             -----------    -------    --------------    -----------    ---------    -----------------------
                           ----
cubbyhole/    cubbyhole    cubbyhole_4fee66ca    n/a           n/a        false             local          false        false
t storage                  ad13b9d2-a4fe-210d-cdc2-dfc9a6fc8c8e
identity/     identity     identity_9033b048     system        system     false             replicated     false        false
                           b2bd00a3-40b1-b9bb-9ce9-ef2942da6f67
kv/           kv           kv_65b023ce           system        system     false             replicated     false        false
                           e62f0380-63fe-dd36-895f-454e2fb3af25
sys/          system       system_873901c7       n/a           n/a        false             replicated     false        false
or control, policy and debugging    556161bf-239e-427b-b591-a8001341b314
transit/      transit      transit_d6f3bbb2      system        system     false             replicated     false        true
                           43497184-016e-c16c-d974-258eea154f75
```

3. Notice that the **External Entropy Access** is set to **true** for **transit/**.

4. Use the `transit` secrets engine to encrypt sensitive data that leverages the HSM as its external entropy source. Create a new encryption key named "orders".

```
# ./vault write -f transit/keys/orders

Success! Data written to: transit/keys/orders
```

5. Send a base64-encoded string to be encrypted by Vault.

```
# ./vault write transit/encrypt/orders plaintext=$(base64 <<< "4111 1111 1111
1111")

Key            Value
---            -----

ciphertext
vault:v1:ZXKU9Yc8+BefMCkPJVUksh5y0NlTymeToyTKl7NzdE5I4CpRtcjjPnUsvVmwPpQ/
```

6. Verify that the ciphertext can be decrypted.

```
# ./vault write transit/decrypt/orders
\ciphertext="vault:v1:ZXKU9Yc8+BefMCkPJVUksh5y0NlTymeToyTKl7NzdE5I4CpRtcjjPnUsv
VmwPpQ/"

Key           Value
---           -----

plaintext     NDExMSAxMTExIDExMTEgMTExMQo=
```

7. Decode to get the original plaintext string.

```
# base64 --decode <<< NDExMSAxMTExIDExMTEgMTExMQo=

4111 1111 1111 1111
```

NOTE: When the external entropy access is enabled, the connectivity to the HSM is required. If the HSM becomes unreachable for any reason, the transit secrets engine returns an error and data cannot be encrypted or decrypted until the HSM connection gets restored.

This completes the integration of HashiCorp Vault with Luna HSM or DPoD Luna Cloud HSM service.

Rotating HashiCorp Vault Keys

The PKCS11 seal supports rotating keys by using different key labels to track key versions. To rotate the key value, you generate a new key in a different key label in the HSM and update Vault's configuration with the new key label value. Restart your Vault instance to pick up the new key label and all new encryption operations will use the updated key label. Old keys must not be disabled or deleted as they are used to decrypt older data. If rotation is desired for data that was seal wrapped prior to this version, set `default_key_label` and `hmac_default_key_label` to allow for decryption of older values.

To rotate HashiCorp Vault Keys:

1. Stop the Vault server from the terminal, if running.
2. Change the configuration file `config.json` as follows:

```
#Entropy
entropy "seal" {
mode = "augmentation"
}

# PKCS11 seal
seal "pkcs11" {
  lib = "<path to cryptoki library>"
  slot = "<slot number>"
  pin = "<partition password>"
  default_key_label="HashiCorp"
  key_label = "HashiCorp_rot"
  default_hmac_key_label = "HashiCorp_hmac"
  hmac_key_label = "HashiCorp_hmac_rot"
  generate_key = "true"
}
storage "file" {
  path = "/tmp/vault"
}
```

3. Start the Vault using the updated configuration file.

   ```
   # ./vault server -config config.json
   ```

4. Launch a new terminal session and execute the following command :

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

5. Verify the Vault status and list the secrets by executing:

```
# ./vault secrets list
```

```
[root@hashicorp]# ./vault secrets list
Path          Type         Accessor              Description
----          ----         --------              -----------
cubbyhole/    cubbyhole    cubbyhole_16b25559    per-token private secret storage
identity/     identity     identity_a6ee82e3     identity store
kv/           kv           kv_ae934885           n/a
secret/       kv           kv_70be3e30           key/value secret storage
sys/          system       system_494262c6       system endpoints used for control, policy and debugging
```

6. Verify the keys generated in the partition by executing `partition contents` in lunacm utility on host.

```
lunacm:>partition contents

        The 'Crypto Officer' is currently logged in.  Looking for objects
        accessible to the 'Crypto Officer'.

        Object list:

        Label:         hashicorp_hmac_rot
        Handle:        102
        Object Type:   Symmetric Key
        Object UID:    37000000180000025b990800

        Label:         hashicorp_rot
        Handle:        99
        Object Type:   Symmetric Key
        Object UID:    36000000180000025b990800

        Label:         hashicorp_hmac
        Handle:        97
        Object Type:   Symmetric Key
        Object UID:    35000000180000025b990800

        Label:         hashicorp
        Handle:        85
        Object Type:   Symmetric Key
        Object UID:    34000000180000025b990800


        Number of objects:  4


Command Result : No Error
```

This completes the rotation of the HashiCorp Vault keys using Luna HSM or DPoD Luna Cloud HSM service.

Contacting Customer Support

If you encounter a problem while installing, registering, or operating this product, refer to the documentation. If you cannot resolve the issue, contact your supplier or Thales Customer Support. Thales Customer Support operates 24 hours a day, 7 days a week. Your level of access to this service is governed by the support plan arrangements made between Thales and your organization. Please consult this support plan for further information about your entitlements, including the hours when telephone support is available to you.

Customer Support Portal

The Customer Support Portal, at https://supportportal.thalesgroup.com, is a database where you can find solutions for most common problems. The Customer Support Portal is a comprehensive, fully searchable repository of support resources, including software and firmware downloads, release notes listing known problems and workarounds, a knowledge base, FAQs, product documentation, technical notes, and more. You can also use the portal to create and manage support cases.

> **NOTE:** You require an account to access the Customer Support Portal. To create a new account, go to the portal and click on the **REGISTER** link.

Telephone Support

If you have an urgent problem, or cannot access the Customer Support Portal, you can contact Thales Customer Support by telephone at +1 410-931-7520. Additional local telephone support numbers are listed on the support portal.

Email Support

You can also contact technical support by email at technical.support.DIS@thalesgroup.com.