COHESITY

Version 1.0

March 2024

Manage Cohesity Encryption Keys Using HashiCorp Vault 1.4

ABSTRACT

To ensure the security of your data, both in flight and at rest, Cohesity supports AES-256 software encryption using an internal Key Management Service (KMS) that automatically generates keys and stores them internally. However, if you manage your encryption keys in a centrally managed external KMS like HashiCorp Vault, you can configure Cohesity DataPlatform to use that instead. Use this guide to set up HashiCorp Vault KMIP 1.4 as your external KMS in Cohesity.

Table of Contents

Figures

KEK Management Overview

As organizations work to address a stream of security threats to the data they manage and store, encryption becomes more critical to every aspect of their cyber defense strategies. Among its many security features, Cohesity supports using an external key management system (KMS). This guide gives you the information and procedures to configure and integrate Cohesity DataPlatform™ and HashiCorp® Secrets Management.

When used with an external key manager like HashiCorp Vault, the Cohesity cluster requests that the key manager creates a Key Encryption Key (KEK) for each Storage Domain in the cluster. These KEKs are used to encrypt and decrypt the Data Encryption Keys (DEKs) created and stored locally in the cluster. As the name implies, the DEKs help you encrypt and decrypt the data.

The Cohesity cluster reaches out to HashiCorp to retrieve the KEKs after the system reboots or the keychain service restarts. If the HashiCorp vault is unavailable, the data in the Storage Domains remains encrypted and inaccessible.

KMIP and Certificate Requirements

The Key Management Interoperability Protocol (KMIP) facilitates communication between the Cohesity cluster and the key server on the HashiCorp vault. KMIP requires the use of Transport Layer Security (TLS) to provide a secure connection. X.509 certificates must exist for the key server and the Cohesity cluster. When you install HashiCorp, the internal Certificate Authority (CA) automatically generates and signs the key server certificate. You must create a client certificate for Cohesity using a tool like OpenSSL. This certificate may be self-signed or externally signed, but this characteristic must be a system-wide characteristic within the HashiCorp environment. In other words, all KMIP clients connecting to HashiCorp must either use self-signed or externally-signed certificates — they cannot use a mixture of both.

Prerequisites

The procedures in this guide assume a basic knowledge of HashiCorp concepts and licensing requirements for KMIP functionality, as well as:

- Cohesity DataPlatform version 6.8.1 or later is installed and operational, and the cluster is configured to use encryption. You can only enable cluster-level encryption when you create the Cohesity cluster. See the Setup Guide for your specific cluster model in the online Help.

  NOTE: When encryption is enabled at the cluster level in Cohesity DataPlatform, it cannot be disabled later.

- Cohesity release version 6.8.1.
- HashiCorp KMIP version 1.4.

Product/Solution	Feature	Application	Functionality
HashiCorp Vault v1.13.2+ent, built 2023-04-25T19:30:45Z	KMIP1_4	raft (HA available)	KMS

- HashiCorp is installed and operational and is accessible by the Cohesity cluster on port 5696. For releases before Cohesity DataPlatform version 6.5, use the Linux 'firewall command' to enable firewall ports on each node. For 6.5 and later, see Manage Application-Based Firewall Rules in the online Help.
- You know your organization's security policies governing data-at-rest encryption.
- You have access to OpenSSL or some other mechanism for generating a client certificate and private key in the Privacy Enhanced Mail (PEM) format on a Linux machine.
- HashiCorp can be used for encrypting Storage Domains and External Targets.

Encryption and Configuration Considerations

The following are some key points to understand regarding this integration:

- Once you enable encryption on a Cohesity cluster, you cannot disable it.
- Once you configure a Cohesity cluster to use an external Key Management System (KMS), you cannot return to using the internal KMS.
- With encryption enabled at the cluster level, all Storage Domains in the cluster are encrypted by rule.
- The Cohesity cluster supports the configuration of one external KMS, and the IP address of the KMS cannot be altered once configured.
- Once it establishes a TLS connection with HashiCorp Vault, a Cohesity cluster never tears down that connection. This results in persistent TLS connections.

Set Up Cohesity-HashiCorp Integration

There are several tasks involved in deploying the Cohesity integration with HashiCorp Vault:

- Use OpenSSL to create a private key and client certificate for the Cohesity cluster.
- Use OpenSSL to extract the internal root CA certificate from HashiCorp.
- Configure HashiCorp DSM to support the integration, including creating a host for the Cohesity cluster. The client certificate will be imported into the host.
- Configure the Cohesity cluster to use HashiCorp as its external Key Management System (KMS KMIP 1.4).

Create the Client Certificate and Private Key

The first step is to create a client certificate and private key for Cohesity that you will use when you configure HashiCorp Vault with the Cohesity cluster. You can use OpenSSL or any similar tool to create them, as long as the certificate and key files are in the PEM format and created on a Linux machine.

There are two different types of client certificates:

- Self-signed certificates. If your security policy allows for it, you can generate and sign your client certificate.
- Externally signed certificates. If your security policy calls for an external Certificate Authority (CA), use this option. To use it, you must have:
  - HashiCorp Vault configured to support system-wide use of externally signed certificates.
  - Established the external CA as trusted by importing the necessary CA certificates.
  - An external, trusted CA that is available to sign the Client Signing Request (CSR).

NOTE: The certificate files need to be available on the machine where you plan to log in to Cohesity DataPlatform to configure the KMS settings.

Generate a Self-Signed Certificate and Private Key

To generate a self-signed certificate and private key for Cohesity:

1. Log in to the Cohesity command-line interface (CLI).
2. Use the `genrsa` command to generate the private key that will be written to the key filename and length you specify.

   ```
   $ openssl genrsa -out <key_file_name> <key_length>
   ```

   NOTE: Key lengths less than 2048 bits are not secure.
3. Enter the following OpenSSL command to create the self-signed certificate per your security policy.

   ```
   $ openssl req -new -x509 -key <key_file_name> -out <cert_file_name> -days
   <validity_period>
   ```
4. Enter the requested information when prompted by OpenSSL:
   - **Country Name**. Your two-letter country code.
   - **State or Province Name**. The full name of your state.
   - **Locality Name (eg, city)**. The full name of your city.
   - **Organization Name**. The name of your organization.
   - **Organizational Unit Name**. The name of your department.
   - **Common Name**. The name that must match the name of the host created on HashiCorp Vault.
   - **Email Address**. (*Optional*) Your email address.

- **Challenge password**. (*Optional*) Leave this blank; click **Enter**.
- **Company name**. (*Optional*) Leave this blank; click **Enter**.

5. When the process is complete, you can find the generated file in the current directory.

```
[cohesity@virtual-robo-esx ~]$ openssl req -new -x509 -key client_key.key -out
client_cert.crt -days 365
You are about to be asked to enter information that will be incorporated into your
certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:US
State or Province Name (full name) []: California
Locality Name (eg, city) [Default City]:San Jose
Organization Name (eg, company) [Default Company Ltd]:Cohesity
Organizational Unit Name (eg, section) []:Engineering
Common Name (eg, your name or your server's hostname) []:dsm.cohesity.com
Email Address []:
[cohesity@virtual-robo-esx ~]$
```

6. You must generate the file on a Linux machine, and if you're creating it manually, be careful that the file doesn't contain any extraneous characters.

```
[cohesity@virtual-robo-esx ~]$ cat client_cert.crt
-----BEGIN CERTIFICATE-----
MIIDxTCCAq2gAwIBAgIJAOSurkIpu6d4MA0GCSqGSIb3DQEBCwUAMHkxCzAJBgNV
BAYTAIVTMRMwEQYDVQQIDApDYWxpZm9ybmIhMREwDwYDVQQHDAhTYW4gSm9zZTER
MA8GAlUECgwIQ29oZXNpdHkxFDASBgNVBAsMC0VuZ2IuZWVyaW5nMRkwFwYDVQQD
DBBkc20uY29oZXNpdHkuY29tMB4XDTIwMDQxNzE3MDcwNVoXDTIxMDQxNzE3MDcw
NVoweTELMAkGAlUEBhMCVVMxEzARBgNVBAgMCkNhbGImb3JuaWExETAPBgNVBAcM
CFNhbiBKb3NIMREwDwYDVQQKDAhDb2hIc2I0eTEUMBIGAlUECwwLRW5naW5IZXJp
bmcxGTAXBgNVBAMMEGRzbS5jb2h!c2I0eS5jb20wggEiMA0GCSqGSIb3DQEBAQUA
A4IBDwAwggEKAoIBAQDMwMq7EW+ikiKZTyhvR0G2+/lJTy0HrI6bzyo3PRDXC+Mb
67yX3sCZDuAodY5bblXiyRoZrxFjwHi6xu0+mYedPhMFHSdo9CPQF2gx2GfWZtEB
FH0qZFZpXug3evqz2IOKBj8SLtFto4uKxSE9EfkMM/IezcP8JVsW7cERohuqc38V
BuLU7LH52vKH+oY0P5v0vpp8IPMJ2zB5vVSh97NTiF/Yo4sWBxeLBQfGmQjxLqkp
zSKGt7T3cle5d4AgPgITzLs6ia+XGQHvoZyIuYnD797hgoJfJZ2R82b8zJRms7IG
et8TXn!cic6+067It2z8fSveyBfRnDyphyWvKhpnAgMBAAGjUDBOMBOGAlUdDgQW
BBS74xB8ZVcniPRZgErlx601kO9+8jAfBgNVHSMEGDAWgBS74xB8ZVcniPRZgEr1
x601kO9+8jAMBgNVHRMEBTADAQH/MA0GCSqGSIb3DQEBCwUAA4IBAQByOZndS7Y8
tG2a2HAzPZB6p9DBDvpNGtT+kpXSjh5uoU4PdB2//k3zGr9UIE8TYBFbLlTv!tMn
908EVcmlbSYjnK+4k9LWOd0PWsLn4a9mFiTvIibatQ3CkjYZAv8jg2y6GI01s6wC
xvWun70jkTxjf2S4bT0ioI+jPp/p0d9oafa+AWAcIMieEm2Gh7n75dZRWQ6Ne7XI
xKk61kASKBiurnzRSvgaCBTTcfTG0fVhZWE7FFAf0C2vAMEFgiI8H65Lgw9f4Lmo
dngrIrD3JQms93QymIeyoF+wwrDsl0nOxONrgJ3oAEH+VdW37YSCR3WeTq46FW2e
5wBVA5TJIr9z
-----END CERTIFICATE-----
[cohesity@virtual-robo-esx ~]$
```

7. Replace the Cohesity cluster's current SSL certificate with the new self-signed certificate you created in the previous steps:

   a. Start the Cohesity CLI using the following command:

   ```
   cohesity_shell# iris_cli admin@198.51.100.12>
   ```

   b. When prompted, enter the Username and password you use to log into the Cohesity Cluster's User Interface. Once the password is successfully authenticated, the Cohesity CLI console opens.
   Replace the certificate by running the following command:

   ```
   cluster update-ssl-certificate ssl-certificate=<absolute path of
   the cert.pem file> ssl-cert-private-key=<absolute path of the
   key.pem file>

   Example:
   admin@127.0.0.1>cluster update-ssl-certificate ssl-
   certificate=/home/support/cert.pem ssl-cert-private-
   key=/home/support/key.pem
   ```

   c. Restart the UI and REST API Services using the following command:

   ```
   admin@198.51.100.12>cluster restart service-names=iris
   ```

   Wait for a minute before proceeding with the next command.

   d. Restart the I/O Operations service using the following command:

   ```
   admin@198.51.100.12>cluster restart service-names=bridge
   ```

Reference: Update SSL Certificates.

Generate an Externally Signed Certificate and Private Key

If you choose to use an externally signed certificate instead of the self-signed certificate, you need to do so with an external and trusted Certificate Authority.

To generate an externally signed certificate and private key:

1. Log in to the Cohesity DataPlatform command-line interface (CLI).
2. Use the `genrsa` command to generate the private key that will be written to the key filename and length you specify. Create the key per your organization's security policy.

```
$ openssl genrsa -out <key_file_name> <key_length>
```

For example:

3. Enter the following OpenSSL command to initiate the CSR file generation process.

```
$ openssl req -new -key <key_file-name> -out <csr_file_name>
```

4. Enter the requested information as prompted by OpenSSL:
   - **Country Name**. Your two-letter country code.
   - **State or Province Name**. The full name of your state.
   - **City Name**. The full name of your city.
   - **Organization Name**. The name of your organization.
   - **Organizational Unit Name**. The name of your department.
   - **Common Name**. The name that must match the name of the host created on HashiCorp Vault.
   - **Email**. (*Optional*) Your email address.
   - **Challenge password**. (*Optional*) Leave this blank; click **Enter**.
   - **Company name**. (*Optional*) Leave this blank; click **Enter**.
5. When the process is complete, you can find the generated CSR file in the current directory.

```
[cohesity@virtual-robo-esx ~]$ openssl req -new -key cohesity241.key -out
cohesity241.csr
You are about to be asked to enter information that will be incorporated into your
certificate request.
What you are about to enter is what is called a Distinguished Name or a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [XX]:US
State or Province Name (full name) []: California
Locality Name (eg, city) [Default City]:San Jose
Organization Name (eg, company) [Default Company Ltd]:Cohesity
Organizational Unit Name (eg, section) []:Engineering
Common Name (eg, your name or your server's hostname) []:dsm.cohesity.com
Email Address []:
[cohesity@virtual-robo-esx ~]$
```

6. Have a trusted CA sign the CSR and download or create a file that contains the certificate.

```
[cohesity@virtual-robo-esx -]$ vi Hashi_root.crt
[cohesity@virtual-robo-esx -]$ cat Hashi_root.crt
-----BEGIN CERTIFICATE-----
MIIENjCCAx6gAwIBAgIGALOmLIL9MA0GCSqGSIb3DQEBDAUAMIGjMSQwIgYDVQQD
ExtDRyBDQSBTIG9uIGRzbS5jb2hlc2l0eS5jb20xETAPBgNVBAsTCGNvaGVzaXR5
MREwDwYDVQQKEwhjb2hlc2l0eTEQMA4GAlUEBxMHU2FuSm9zZTELMAkGAlUECBMC
Q0ExKTAnBgkqhkiG9w0BCQEWGmZlcWlhbmcuemhhbmdAY29oZXNpdHkuY29tMQsw
CQYDVQQGEwJVUzAeFw0yMDA0MTUyMzQ5MThaFw0zMDA0MTcyMzQ5MThaMIGjMSQw
IgYDVQQDExtDRyBDQSBTIG9uIGRzbS5jb2hlc2l0eS5jb20xETAPBgNVBAsTCGNv
aGVzaXR5MREwDwYDVQQKEwhjb2hlc2l0eTEQMA4GAlUEBxMHU2FuSm9zZTELMAkG
AlUECBMCQ0ExKTAnBgkqhkiG9w0BCQEWGmZlcWlhbmcuemhhbmdAY29oZXNpdHku
Y29tMQswCQYDVQQGEwJVUzCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEB
AL5BAxvndW/VoHfpMQPFqw9nzvxddT3yfypbl07TZOeby7C2XHhuIneUzPPr7m/v
C8cTS0eqj75cj0SVwVkMzKiU2nHXxX3qAztnkHhlo5uryWsQk2Kls0fB3GecvjU2
5oDo9NOJqDoU29wKhjgZMAMv/SPIHYu1IfWT0CV5FfjtsdsDYINHiLdUxZIQJF42
r0aPQT0weiTG0V9bJj/ilA2hCUTyVVwXc4erEsDbeT9Sc7kKJ3QEGcBMcVtit5zM
AK5HhMq8T+oS9dhIyWBZpiAVp/rYE2ZfIU2FeplHJtf60zCpIgHr7N12lqviTbsK
svhyLt20teadRitSRlXGweMCAwEAAaNuMGwwEgYDVR0TAQH/BAgwBgEB/wIBADAO
BgNVHQ8BAf8EBAMCAQYwHQYDVR0OBBYEFKgZMHRJBV1MRLi+xHdp0vrjsQSwMCcG
AlUdIwQgMB6AFKgZMHRJBV1MRLi+xHdp0vrjsQSwggYAs6Ysgv0wDQYJKoZIhvcN
AQEMBQADggEBAEsQ29HfaHpHUFBj4Tn74XW0b9w9Pmk2u5AM13d9CU/oLYcMJn0L
tfavpnMgKPuDwQ/3gqgFDXiq40dq73fD+38A2wlcB/xZumQ6nGGuYGk2p2VVy6Cr
JlVD0pL4iwidQXMYJ7xDmL+TwHM8clwPiwxqI3HRNRQokqwl489VeptNMteInllM
IY28TZOn/T89OxUDxoqxIlXEu672qZq7kG/YpeeGG+0uwH8QZnxjogaIQPulTxJs
5tzcsFM7nRjE63Z14GUIsQElpXikSu9G8ljpGBmkfcJdaVIHSeRTB7NV5ZIj2NvA
UXm/RFvjTxh9blR5yTkdPUDEDgFdJWVA6Fw=
-----END CERTIFICATE-----
[cohesity@virtual-robo-esx -]$
```

7. Replace the Cohesity cluster's current SSL certificate with the new self-signed certificate you created in the previous steps:

   a. Start the Cohesity CLI using the following command:

```
cohesity_shell# iris_cli admin@198.51.100.12>
```

   b. When prompted, enter the Username and password you use to log into the Cohesity Cluster's User Interface. Once the password is successfully authenticated, the Cohesity CLI console opens.
   Replace the certificate by running the following command:

```
cluster update-ssl-certificate ssl-certificate=<absolute path of
the cert.pem file> ssl-cert-private-key=<absolute path of the
key.pem file>

Example:
admin@127.0.0.1>cluster update-ssl-certificate ssl-
certificate=/home/support/cert.pem ssl-cert-private-
key=/home/support/key.pem
```

c. Restart the UI and REST API Services using the following command:

```
admin@198.51.100.12>cluster restart service-names=iris
```

It is recommended to wait a minute before proceeding with the next command.

d. Restart the I/O Operations service using the following command:

```
admin@198.51.100.12>cluster restart service-names=bridge
```

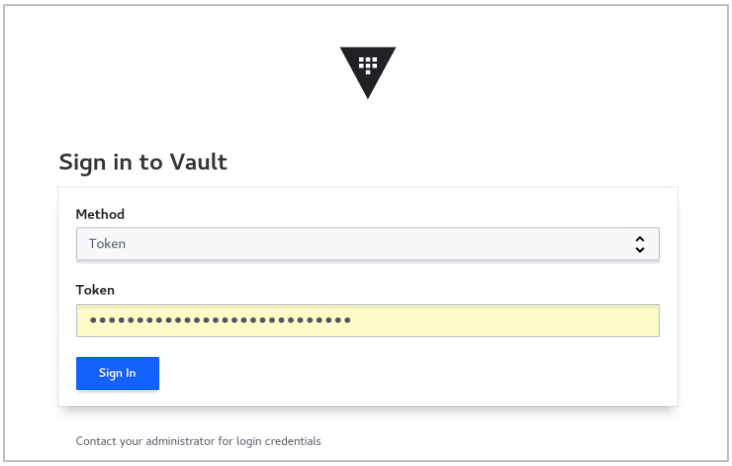
Reference: Update SSL Certificates.

Extract the HashiCorp Vault Internal CA Certificate

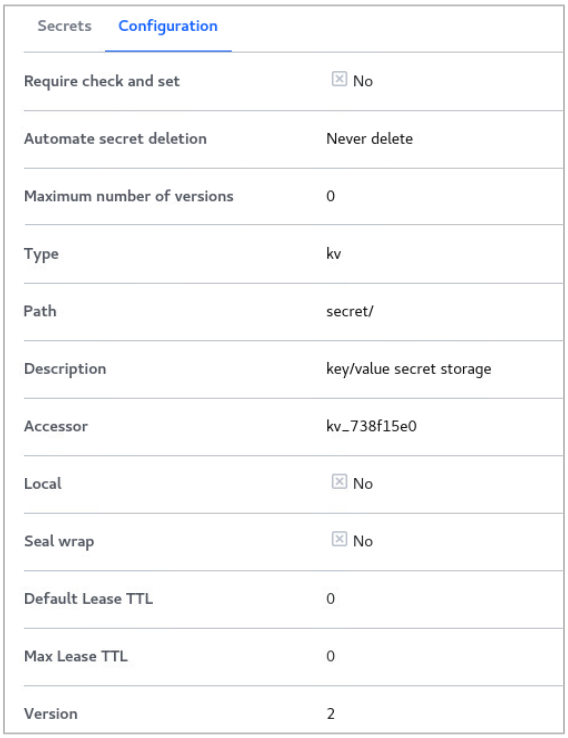
Once you have generated the client certificate and private key, you need to extract the HashiCorp Vault internal root CA certificate for import into the Cohesity cluster. You can extract it via a web browser on a Linux machine or by logging into the HashiCorp GUI.

To extract the HashiCorp internal root CA certificates:

1. Login to HashiCorp Vault GUI HashiCorp Vault GUI.
2. Sign-in to the HashiCorp Vault GUI and obtain the certificates (Client certificate, Client key and CA certificate) using KMIP Secrets Engine:
   a. Login to the vault GUI:



   b. Navigate to the Secrets KMIP configuration:



c. Obtain the keys (Client certificate, Client key and CA certificate):

3. Review the KMIP secrets engine configuration from the CLI session of the HashiCorp server:

```
# vault read kmip/config
Key                            Value
---                            -----
default_tls_client_key_bits    256
default_tls_client_key_type    ec
default_tls_client_ttl         336h
listen_addrs                   [0.0.0.0:5696]
server_hostnames               [localhost]
server_ips                     [127.0.0.1 ::1]
tls_ca_key_bits                256
tls_ca_key_type                ec
tls_min_version                tls12
```

NOTE: When the KMIP engine is configured, it generates and stores a key pair and uses it to sign a root CA which is used to sign an intermediate CA. The intermediate CA is used to sign the server TLS certificate the KMIP listener uses to identify itself during the TLS handshake.

As necessary, transfer the CA certificate file, the client private key file, and the client certificate file to the machine that will be used to authenticate Cohesity DataPlatform and HashiCorp Vault KMIP 1.4 to perform the configuration steps.

Configure HashiCorp Vault KMIP 1.4 Settings

Equipped with your client certificate, private key, and the HashiCorp Vault internal CA certificate, you are ready to configure the HashiCorp Vault settings to support Cohesity for a Key Management Interoperability Protocol (KMIP) integration.

To configure HashiCorp Vault:

1. Log in to your account on the HashiCorp server.
2. Install HashiCorp vault: Install Vault.
3. Configure HashiCorp vault: Configure Vault.
4. Activate Vault UI: Vault UI.
5. Extract the CA certificate, client private key, and the client certificate: KMIP Secrets Engine.
6. Vault Commands CLI: Commands.
7. Vault Troubleshooting: Troubleshooting.

HashiCorp Vault is now configured to support KMIP communication with the Cohesity cluster. You are ready to configure the KMS settings in Cohesity DataPlatform.

Configure Cohesity Key Management Settings

The final step is configuring the KMS settings on the Cohesity cluster. Once the configuration has been saved, the cluster establishes a TLS session with HashiCorp Vault. After a few minutes of internal processing, it requests that keys be created for internal use and the existing Default Storage Domain.

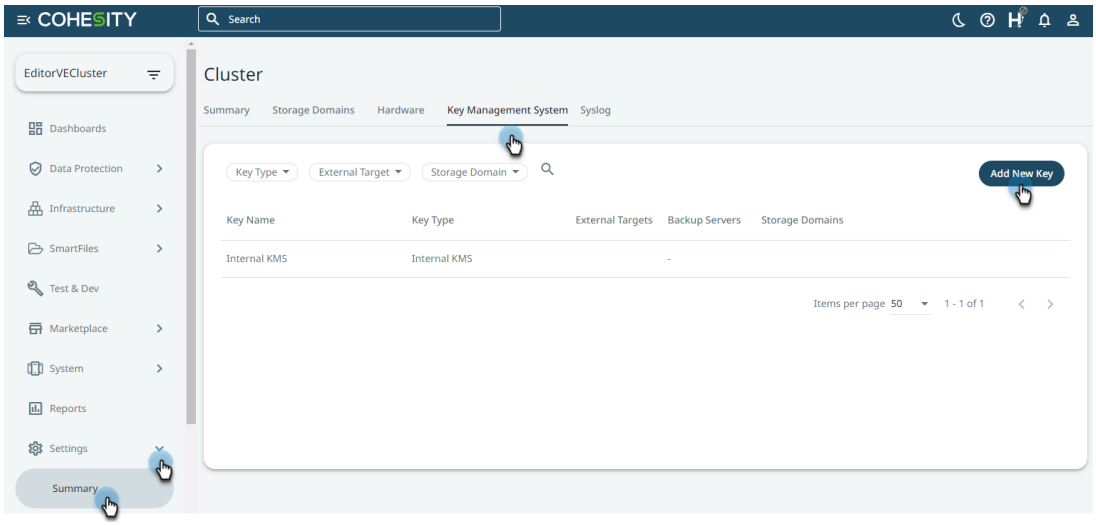
NOTE: The HashiCorp Vault IP address cannot be changed once it is configured.

You can configure HashICorp Vault in the Cohesity DataPlatform browser UI or the Cohesity DataPlatform command-line interface (CLI).

Configure HashICorp Vault in Cohesity DataPlatform UI

To configure HashiCorp Vault in the Cohesity UI:

1. Log in to Cohesity DataPlatform.
2. Navigate to **Settings> Summary**.
3. On the **Cluster Summary** page, click the **Key Management System** tab and click **Add New Key**.

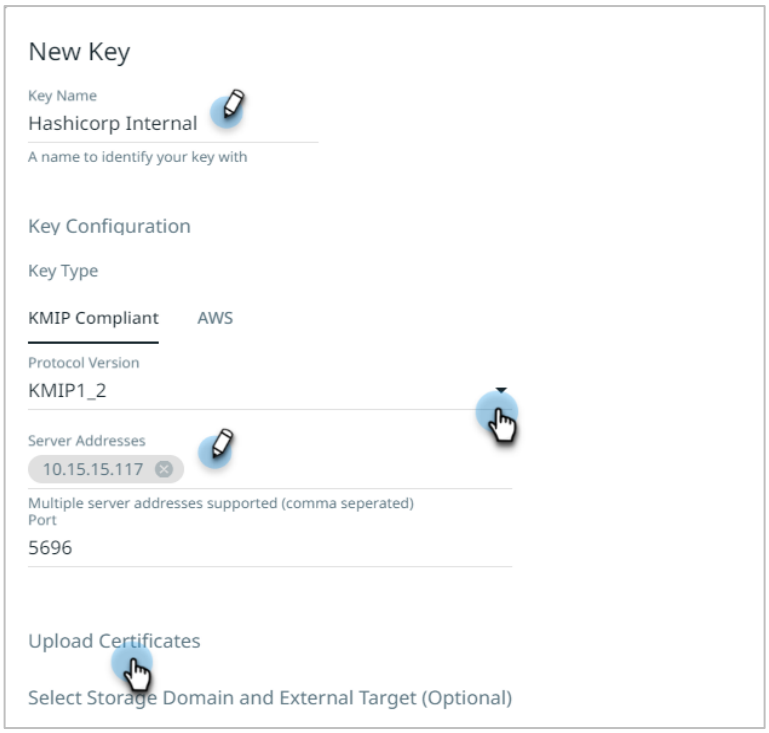


4. In the **New Key** form, type a key name for your Hashicorp key in the Key Name box.**Key Name**.
5. Click and expand the **Key Configuration** tab. Under KMIP Compliant:
   - **Protocol Version**. The version of KMIP to be used.

     NOTE: Acceptable options are: KMIP1_1, KMIP1_2, or KMIP1_3. In our testing, Cohesity used KMIP1_4 for HashiCorp KMIP 1.4.

   - **Server Addresses:** Type the HashiCorp's IP address.
   - **Port**: Keep the default port 5696.

6. Click and expand the **Upload Certificates** tab.
   - **Client Certificate**. Select the client certificate file that you created above.
   - **Client Key**. Select the private key file that you created.
   - **CA Certificate**. Select the CA certificate file that you extracted.
   - **Select Storage Domain and External Target** (Optional).



7. Click **Save**.

The Cohesity cluster immediately attempts to establish a TLS session with HashiCorp Vault and initiate KMIP communication. If you get a **KMS Validation Error**, see Troubleshooting below.

Configure HashiCorp Vault in Cohesity DataPlatform CLI

You can also configure HashiCorp Vault in the Cohesity DataPlatform command-line interface (CLI). See Using the Cohesity DataPlatform CLI in the online Help for the full list of commands.

To configure HashICorp Vault in the Cohesity DataPlatform CLI:

1. SSH to the cluster using the following command:

```
ssh cohesity@<ip address of node>
```

2. In the CLI, use the **kms create** command.

```
admin@x.x.x.x> kms create help
  DESCRIPTION
    To create a new KMS.

Example: kms create ca-certificate="<absolute path to CA certificate>" client-
certificate="<absolute path to client certificate>" dien
t-key="<absolute path to client-key>" kmip-protocol-version="KMIPl_2" kms-
name="KmipServerl" kms-port=5696 kms-ip="IP address of KMS"

  PARAMS
    ca-certificate         [string]  required     File path to ca-certificate.
    client-certificate     [string]  required     File path to client-certificate.
    client-key             [string]  required     File path to client-key.
    kmip-protocol-version [string]  required     kmip-protocol-version
    kms-ip                 [string]  required     IP address of the KMS.
    kms-name               [string]  required     Name of the KMS.
    kms-port               [int]     required     KMS Port. Default KMIP port is
5696.

admin@x.x.x.x>
```

3. After successfully connecting with HashICorp Vault, Cohesity automatically creates one key for a default Storage Domain and other keys for internal use.

Figure 1: KMS Configuration

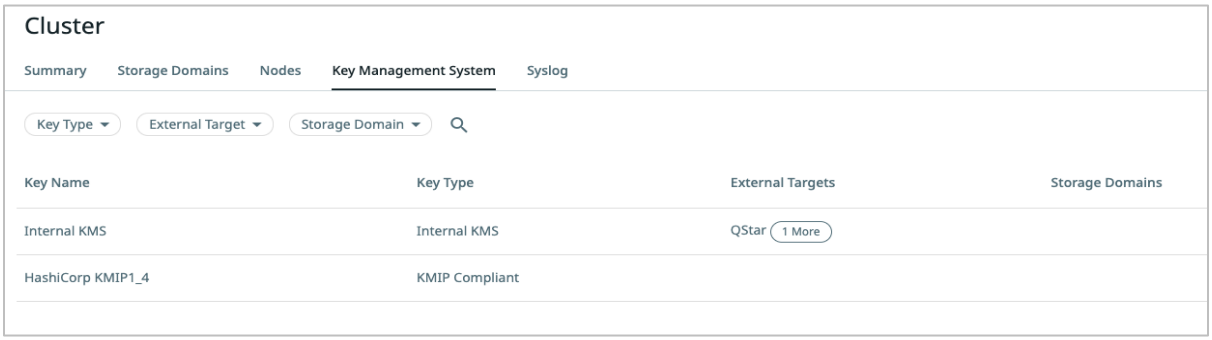
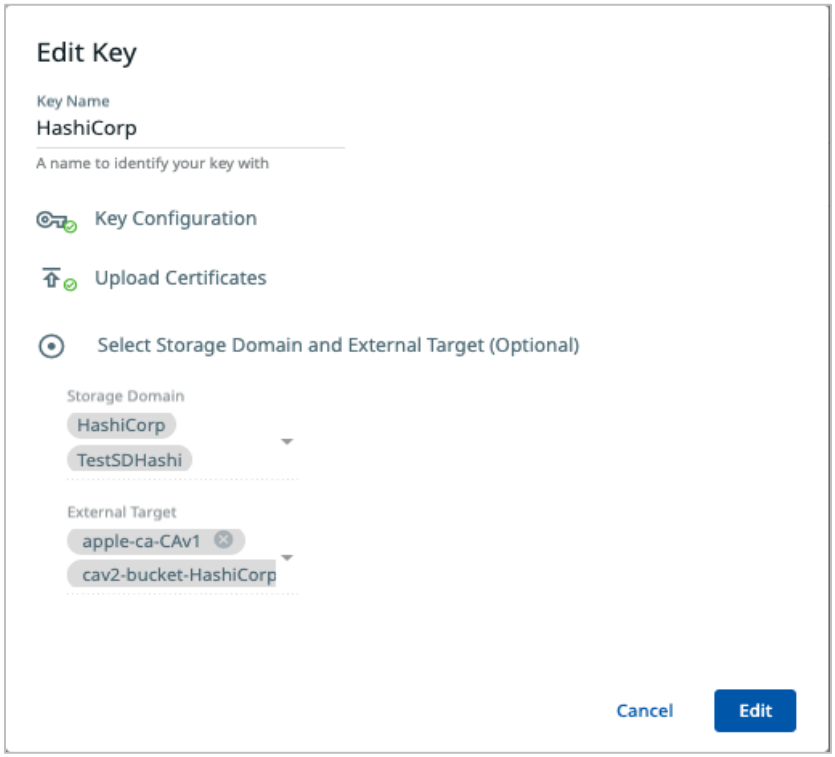


Figure 2: Edit KMS Key and configure Storage Domain and External Target



If you see the above keys in the GUI, it means that the KEKs have migrated from your Cohesity cluster to the HashiCorp vault. This completes the integration of HashiCorp Vault with the Cohesity cluster:

Update Cohesity Key Management Configuration

If your KMS configuration has changed because of expired certificates or a change in KMIP protocol version, you can update the Key Management System settings in Cohesity DataPlatform. As with the initial Cohesity KMS configuration, you can update these in the Cohesity DataPlatform browser UI or in the Cohesity DataPlatform CLI.

Update Cohesity KMS in Cohesity DataPlatform UI

To update the Cohesity KMS settings using the browser UI:

1. Navigate to **Platform** > **Cluster** and click the **Key Management System** tab.
2. On the **Key Management System** page, you can edit the configured KMS details like **Server Name**, **Protocol Version**, **Server IP**, and the associated certificates. When you are done, click **Save.**

Update Cohesity KMS in Cohesity DataPlatform CLI

You can also update the Cohesity KMS settings in the Cohesity DataPlatform CLI. See Using the Cohesity DataPlatform CLI in the online Help for the full list of commands.

To update the Cohesity KMS settings using the Cohesity DataPlatform CLI:

1. SSH to the cluster using the following command:

```
ssh cohesity@<ip address of node>
```

2. In the CLI, use the **`kms update`** command.

```
admin@x.x.x.x> kms update help

  DESCRIPTION
    To update an existing KMS.

  PARAMS
    ca-certificate          [string]  optional   File path to ca-certificate.
    client-certificate      [string]  optional   File path to client-certificate.
    client-key              [string]  optional   File path to client-key.
    kmip-protocol-version  [string]  optional   kmip-protocol-version
    kms-ip                  [string]  required   IP address of the KMS.
    kms-name                [string]  optional   Name of the KMS.
    kms-port                [int]     optional   KMS Port. Default KMIP port is
5696.

admin@x.x.x.x>
```

NOTES:

- If at any point after initial configuration you modify the Key Management System settings on the Cohesity cluster, you must manually restart the keychain service. See Keychain Service below.
- The KMS IP address cannot be modified once configured.

Troubleshooting

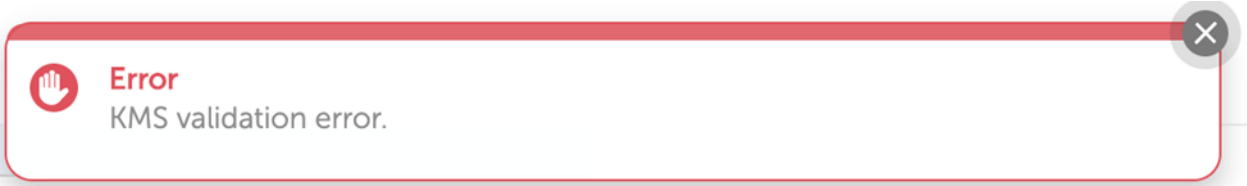
You might encounter errors while configuring KMS or Storage Domain settings in Cohesity DataPlatform. Invalid input parameters or communications errors might cause the error.

The most common errors are:

- A KMS validation error while configuring the KMS.
- A KMS unreachable error while creating a Storage Domain.

KMS Validation Error with KMS Configuration

If the Cohesity cluster cannot communicate with HashiCorp Vault when configuring the Key Management settings, the following generic KMS validation error appears:



If it does, take the following steps:

- Verify correct addressing and basic network connectivity between HashiCorp Vault and the Cohesity cluster.
- Verify port 5696 is configured on the Cohesity DataPlatform KMS settings page and that firewalls are open for that port.
- If any of the uploaded certificate files or private key file on the Cohesity DataPlatform KMS settings page were created on a Windows system, recreate them on a Linux system.

  IMPORTANT: The Cohesity KMS client only accepts an SSL certificate that contains a Unix-style newline character, which is '\n'. Format your certificates accordingly — in Windows, replace '\r\n' with '\n' and on Mac OS, replace '\r' with '\n' — and then load the certificates.
- Verify that the CA certificate uploaded on the Cohesity DataPlatform KMS settings page is the internal root CA certificate from HashiCorp Vault. It should *not* be HashiCorp Vault's server certificate. The Cohesity cluster needs the root CA certificate to validate the server certificate that is delivered to it while establishing a TLS session.
- If you are using an externally signed client certificate, ensure all CA chain certificates have been imported into HashiCorp Vault via the **System > KMIP Trusted CA Certificates** page.
- If you are using a self-signed client, ensure there are no KMIP Trusted CA Certificates imported into HashiCorp Vault. If there are, you'll have to create an externally signed client certificate that has been signed by one of those trusted CAs.

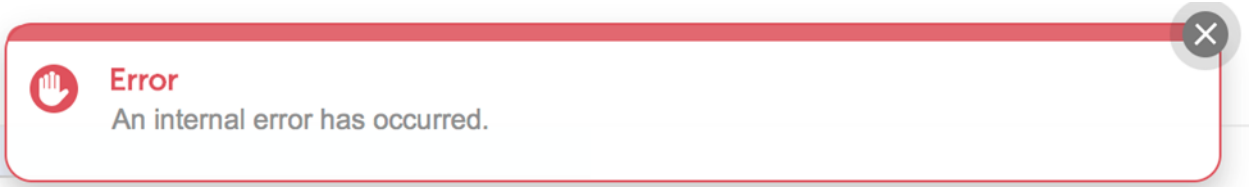
- Verify the host configuration on HashiCorp Vault for the Cohesity cluster:
  - The **Host Name** must match the **Common Name** in the client certificate.
  - The client certificate must be uploaded into the host configuration on HashiCorp Vault, and its **Certificate Fingerprint** should be visible on the host configuration page.
  - The KMIP **Registration Allowed** and **Communication Enabled** settings should be checked.
- Proper licensing must be in place.

KMS Unreachable Error with Storage Domain Creation

When you create a new Storage Domain, the Cohesity cluster immediately sends a key generation request to HashiCorp Vault. If a TLS session is not established or if HashiCorp Vault is unreachable, the Storage Domain will not be created, and you will see the following error:

A possible cause of this error is that the TLS session with HashiCorp Vault has been dropped due to inactivity. The Cohesity cluster will immediately take action to re-establish the connection, but not before you see the ***KMS is unreachable*** error message. To remedy this, simply click the **Create Storage Domain** button to try again. If the problem was a dropped TLS session, it should work the second time.

Suppose the problem was not just the lack of a TLS session, and there is indeed a connectivity issue of some type. In that case, you will either continue to see the ***KMS is unreachable*** error or possibly the ***internal error*** message below. To resolve this, try the steps in KMS Validation Error above.



To resolve this, try the steps in KMS Validation Error above.

Keychain Service

The keychain service on Cohesity DataPlatform is responsible for communicating with HashiCorp Vault.

An error log for the keychain service exists in the `/logs` directory of the Cohesity instance. It is accessible by SSH'ing into the instance to access the Linux OS. The filenames are `keychain_exec.ERROR`, `keychain_exec.INFO`, and `keychain_exec.FATAL`.

Restart the Keychain Service after Key Management Settings Update

If you update the Key Management settings after initially configuring them, you have to restart the keychain service for the new settings to take effect.

To restart the Cohesity keychain service in the Cohesity DataPlatform CLI,

1. Start the Cohesity DataPlatform CLI remotely or locally using the following command.

```
$ iris_cli -server x.x.x.x -username=<username> -password=<password>
```

2. Issue the following command:

```
$ cluster restart service-names="keychain"
```

See Restart the Cohesity Keychain Service in the online Help for more.

Your Feedback

Was this document helpful? Send us your feedback!

About the Authors

Kunal Bose is Sr. Product Solutions Architect, Product Solutions WWFO, at Cohesity. In his role, Kunal focuses on enterprise data protection, solution validation, solution testing, solution qualification, and software usability.

Other essential contributors included:

- Sagar Sethi, Staff Technical Solutions Engineer at Cohesity
- Subash Babu, Staff Technology Editor at Cohesity
- Luke Walker, Lead Product Manager at Cohesity

Document Version History

VERSION	DATE	DOCUMENT HISTORY
1.0	Mar 2024	First release

ABOUT COHESITY

Cohesity is a leader in AI-powered data security and management. Aided by an extensive ecosystem of partners, Cohesity makes it easier to protect, manage, and get value from data – across the data center, edge, and cloud. Cohesity helps organizations defend against cybersecurity threats with comprehensive data security and management capabilities, including immutable backup snapshots, AI-based threat detection, monitoring for malicious behavior, and rapid recovery at scale. Cohesity solutions are delivered as a service, self-managed, or provided by a Cohesity-powered partner. Cohesity is headquartered in San Jose, CA, and is trusted by the world's largest enterprises, including six of the Fortune 10 and 42 of the Fortune 100.

Visit our website and blog, follow us on Twitter and LinkedIn and like us on Facebook.