

## 

## Keyfactor Vault Secrets Engine Guide

The Keyfactor Secrets Engine for Hashicorp Vault is a Vault plugin that replicates Vault’s onboard PKI API and processes certificate enrollment requests through the Keyfactor Command or Keyfactor Control platform. In many cases the onboard PKI engine included with Vault can be swapped for the Keyfactor engine seamlessly, with no impact to Vault client applications. While the simplicity of the onboard PKI is attractive to developers who are trying to implement the simplest solution in order to meet encryption requirements, it presents other enterprise teams with some challenges when it comes to PKI operations and security:

* The Vault infrastructure and root materials are not managed by PKI professionals and policies, but rather by DevOps teams that may not be trained in how to properly handle and manage an enterprise PKI.
* Lack of Certificate Lifecycle Management places organizations in a reactionary posture. If there are weaknesses in the organization processes, full visibility of the certificates is necessary in order to identify these risks prior to a security event or audit failure.
* All certificates are susceptible as an attack surface and should be managed and monitored, regardless of their lifetime, to ensure that issuance policies and certificate standards are followed.

Keyfactor Command can provide the control and visibility needed for a Vault environment. Using the Keyfactor Secrets Engine plugin for Vault, PKI functionality is directed to your enterprise PKI environment, placing control back into the hands of the enterprise PKI admins, and allowing your PKI admins to stay in control of how and when certificates are issued. The Keyfactor Secrets Engine offers the following enterprise capabilities:

* Issue certificates and place them into the Vault secrets store using your existing enterprise PKI.
* Eliminate the need for a standalone PKI within the vault environment.
* Gain complete visibility and management of certificates across all Vault instances and manage them through a single pane of glass.
* Reporting, alerting, automation, and auditing on the certificates within the environment.
* Easily identify and revoke non-compliant or rogue certificates.
* Integrate with SIEMs and ticketing systems for automated notifications.

## Installation

1. Create an API application in Keyfactor to be used by the plugin.
2. Ensure that your Keyfactor instance has the API application setting “Disable HMAC Signature Validation” set to *true (Note: This requirement will be eliminated in a future version of the Keyfactor Secrets Engine)*
3. Create or identify a user in a Security Role with the following permissions:
   1. API: Read
   2. Certificate Enrollment: Enroll CSR
   3. Certificates: Revoke
4. Copy the Keyfactor Secrets Engine binary into your Hashicorp Vault plugins directory
5. Create a JSON file on the machine hosting the engine with the following parameters:

|  |  |
| --- | --- |
| Parameter | Value |
| host | Hostname or IP address of Keyfactor server. |
| creds | Basic auth credentials for the user identified in step 3 (base-64 encoding of “DOMAIN\user:Password”). |
| appkey | Base-64 encoding of the API application key. |
| secret | Base-64 encoding of the API secret key. |
| template | Active Directory certificate template to use for certificate requests. |
| protocol | One of “http” or “https”. For making requests to the Keyfactor server. |
| CA | Distinguished name of the CA that should sign certificate requests. |

Example:

{

"host":"192.168.0.24",

"creds":"SkRLXGpraWxnYWxsOlBAc3N3MHJk",

"appkey":" LBfep9KOH0sNHg==",

"secret": "AbIDf2NUNh41oQ==",

"template":"User",

"protocol":"https",

"CA":"CN=jdk-CA1,DC=jdk,DC=cms"

}

*(Note: Ensure permissions are set appropriately on this file to avoid credential disclosure)*

1. Point the plugin to the configuration file with the KF\_CONF\_PATH environment variable

export KF\_CONF\_PATH=/path/to/json/file

1. Enable the Keyfactor Secrets Engine plugin in your Vault instance

vault write sys/plugins/catalog/keyfactor sha256=47f549d44ab2abcb528aa45725b3a83334a9465bb487f3d1182add55e5580c36 command="keyfactor"

vault secrets enable keyfactor

## Role Management

Create/update role

vault write keyfactor/roles/<rolename> allowed\_domains=<domain1>,<domain2> allow\_subdomains=true

List roles

vault list keyfactor/roles

Read role

vault read keyfactor/roles/<rolename>

Delete role

vault delete keyfactor/roles/<rolename>

## Operation Examples

The following commands correspond to the requests available through the Vault PKI API

Request certificate

vault write keyfactor/issue/<rolename> common\_name=<CN>

List certificates

vault list keyfactor

Read certificate

vault read keyfactor/<Serial Number>

*(Note: Certificate serial numbers are provided in the output for enrollment and list commands)*

Revoke certificate

vault write keyfactor/revoke serial\_number=<serial>

Sign CSR

vault write keyfactor/sign/<rolename> csr=<csr>

Read CA cert

vault read keyfactor/ca

Read CA chain

vault read keyfactor/ca\_chain

Additional notes for this plugin:

* TTL management is not handled through the secrets engine. Validity period is determined by certificate template. Expiration time is not reported in the enrollment output (see example below)

## Example

jdk@ubuntu:~$ vault write keyfactor/roles/jdk allowed\_domains=jdk.cms

Success! Data written to: keyfactor/roles/jdk

jdk@ubuntu:~$ vault write keyfactor/issue/jdk common\_name=jd.jdk.cms

Key Value

--- -----

certificate -----BEGIN CERTIFICATE-----

MIIFqjC…uJNVPfg1g=

-----END CERTIFICATE-----

issuing\_ca -----BEGIN CERTIFICATE-----

MIIDbTC…aM6It0=

-----END CERTIFICATE-----

private\_key -----BEGIN RSA PRIVATE KEY-----

MIIEpgIB…OJadtHXd4UPEM

-----END RSA PRIVATE KEY-----

private\_key\_type rsa

serial\_number 690003E8BAF8B3973C9B61FD2800000003E8BA

jdk@ubuntu:~$ vault list keyfactor

Keys

----

690003E8BAF8B3973C9B61FD2800000003E8BA

jdk@ubuntu:~$ vault read keyfactor/690003E8BAF8B3973C9B61FD2800000003E8BA

Key Value

--- -----

690003E8BAF8B3973C9B61FD2800000003E8BA MIIFqjCCB…fg1g=

jdk@ubuntu:~$ vault delete keyfactor/690003E8BAF8B3973C9B61FD2800000003E8BA

Success! Data deleted (if it existed) at: keyfactor/690003E8BAF8B3973C9B61FD2800000003E8BA