HASHICORP VAULT / KMES SERIES 3 PKCS #12 PASSWORD EXPORT

Integration Guide

Applicable Devices:
KMES Series 3

TABLE OF CONTENTS

[1] OVERVIEW OF THE HASHICORP VAULT / KMES SERIES 3 SECRET EXPORT INTEGRATION

[1.1] ABOUT HASHICORP VAULT

From HashiCorp's online documentation (https://www.vaultproject.io/docs/what-is-vault): "Vault is a tool for securely accessing secrets. A secret is anything that you want to tightly control access to, such as API keys, passwords, or certificates. Vault provides a unified interface to any secret, while providing tight access control and recording a detailed audit log.

A modern system requires access to a multitude of secrets: database credentials, API keys for external services, credentials for service-oriented architecture communication, etc. Understanding who is accessing what secrets is already very difficult and platform-specific. Adding on key rolling, secure storage, and detailed audit logs is almost impossible without a custom solution. This is where Vault steps in."

[1.2] PURPOSE OF THE INTEGRATION

This integration gives users the ability to store PKCS #12 passphrases in HashiCorp Vault automatically after they are generated on the KMES Series 3. The intention is to rid DevOps and developers from the hassles of creating secrets and populating them into Vault in a secure manner when requesting X.509 certificates and key pairs.

[2] FUTUREX CERTIFICATION PROCESS

The Futurex Certification Process is a rigorous and standardized approach to testing and certifying integrations between third-party applications and Futurex's HSMs and key management servers (i.e., KMES Series 3). The certification process is designed to ensure that third-party application integrations are fully tested and validated in a lab environment before they are deployed in a production environment. Futurex's Integration Engineering team implements this process so that customers can have confidence that third-party applications will integrate seamlessly with Futurex's HSMs and KMES Series 3 devices, and that all operations will result in the expected behavior. The certification process involves several steps, including research, testing, troubleshooting, and certification, and is fully documented in an integration guide for each integration. The full process is outlined below:

1. Research the third-party application to gain a general understanding of the solution and the protocol it uses to integrate with an HSM or KMS device (i.e., PKCS #11, Microsoft CNG, JCE, OpenSSL Engine, KMIP).
2. Determine the scope of the third-party application's use of the HSM or KMS device, including the specific functionalities it utilizes (i.e., data encryption, key protection, entropy, etc.).
3. Install and configure the third-party application in a lab environment, where all testing and validation will take place.
4. Establish a connection between the third-party application and the Futurex device, which typically involves configuring TLS certificates and creating roles and identities that the third-party application will use to connect and authenticate to the Futurex device.
5. Initiate a request from the third-party application to the Futurex device, such as generating keys or certificates, encrypting or decrypting data, or other cryptographic functions.
6. If any errors occur during the testing process, the Integration Engineering team will diagnose the issues and take necessary corrective actions. If necessary, the team will also document the error(s) by creating engineering change requests (ECRs) to ensure all issues are addressed and resolved before certification.
7. After any necessary engineering changes have been made, a new end-to-end test will be performed to ensure that all errors have been resolved and that all operations are successful.
8. Certify the integration by creating an integration guide that covers all necessary prerequisites, configurations required in both the third-party application and the Futurex device, and how to test the functionality.

Overall, following these steps helps ensure that the integration between the third-party application and the Futurex device is fully tested and validated, and that any errors or issues are resolved before the integration is certified as fully supported.

[3] PREREQUISITES

Supported Hardware:

- KMES Series 3, version 6.1.3.11 and above, with the *External Secret Storage* license enabled

Other:

- HashiCorp Vault application
- OpenSSL

[4] VAULT SETUP AND CONFIGURATION

[4.1] DOWNLOAD AND INSTALL VAULT

Please refer to HashiCorp's Vault documentation at the following link for instructions on how to download and install the Vault application: https://www.vaultproject.io/docs/install

NOTE: The second installation option at the link above is using a precompiled binary. These binaries can be downloaded at the following url: https://www.vaultproject.io/downloads

To verify Vault is properly installed, run **vault -h** on your system. You should see help output. If you are executing it from the command line, make sure it is on your PATH or you may get an error about Vault not being found.

```
$ vault -h
```

[4.2] CONFIGURE VAULT

Vault uses documented sane defaults so only non-default values must be set in the configuration file.

Create */etc/vault.d* directory.

```
$ sudo mkdir --parents /etc/vault.d
```

Create a Vault configuration file, *vault.hcl*.

```
$ sudo touch /etc/vault.d/vault.hcl
```

Create a unique, non-privileged system user to run Vault.

```
$ sudo useradd --system --home /etc/vault.d --shell /bin/false vault
```

Set the ownership of the */etc/vault.d* directory.

```
$ sudo chown --recursive vault:vault /etc/vault.d
```

Set the file permissions.

```
$ sudo chmod 640 /etc/vault.d/vault.hcl
```

Configure tcp Listeners in the Vault configuration file

The TCP listener configures Vault to listen on a TCP address/port, as shown in the example below.

```
listener "tcp" {
  address = "127.0.0.1:8200"
}
```

The **listener** stanza may be specified more than once to make Vault listen on multiple interfaces. If you configure multiple listeners you also need to specify **api_addr** and **cluster_addr** so Vault will advertise the correct address to other nodes.

The *vault.hcl* configuration file used for demonstration in this guide is shown below. It shows Vault listening on a private interface, as well as localhost.

NOTE: The values defined in the *vault.hcl* file need to be customized for each specific use case (i.e., IPs and ports; file paths to certificates).

NOTE: In this integration guide, the Vault server will be run in "Dev" mode. When deploying Vault in a production setting there are more things to consider (i.e., the storage backend), but the concept of configuring tcp listeners, as described below, still applies in that case. Please refer to Vault's documentation for specifics on how to deploy Vault in a production environment (https://learn.hashicorp.com/tutorials/vault/getting-started-deploy).

```
# Configure the storage backend for Vault
storage "file" {
  path = "/tmp/vault"
}

# Address and port on which Vault will respond to requests from the KMES Series 3
listener "tcp" {
  address = "10.0.5.118:8210"
  tls_disable = false
  tls_cert_file = "/home/bbarrows/Documents/Vault/client-cert.pem"
  tls_key_file = "/home/bbarrows/Documents/Vault/client-privatekey.pem"
}

# Advertise the non-loopback interface
api_addr = "https://10.0.5.118:8210"

# Enable the Vault web UI
ui = true

# Lock process memory pages, preventing them from being swapped to disk
disable_mlock = true
```

Please reference the comments before each block. They explain what each of the defines is doing.

The most critical information to note is that **10.0.5.118** is the IP of the machine that Vault is installed on, and **8210** is the port on which Vault will listen for requests from the KMES Series 3.

An in-depth explanation of how to set up the client TLS certificates is beyond this course's scope, but **there is one crucial thing to note concerning this: The client certificate's common name must match the IP address set in the address define**. Otherwise, the KMES Series 3 will not verify the certificates presented by Vault to the KMES Series 3.

NOTE: cluster_address is not defined in the *vault.hcl* file above because only a single Vault server is being utilized for this demo.

For more information about configuring the Vault configuration file, please refer to Vault's documentation at the following url: https://learn.hashicorp.com/tutorials/vault/configure-vault

[4.3] START THE DEV SERVER

To start the Vault dev server, run:

```
$ vault server -dev -config=/etc/vault.d/vault.hcl
==> Vault server configuration:

             Api Address: https://10.0.5.118:8210
                     Cgo: disabled
         Cluster Address: https://10.0.5.118:8211
              Go Version: go1.14.7
              Listener 1: tcp (addr: "127.0.0.1:8200", cluster address: "127.0.0.1:8201", max_
request_duration: "1m30s", max_request_size: "33554432", tls: "disabled")
              Listener 2: tcp (addr: "10.0.5.118:8210", cluster address: "10.0.5.118:8211", max_
request_duration: "1m30s", max_request_size: "33554432", tls: "enabled")
              Listener 3: tcp (addr: "127.0.0.1:8210", cluster address: "127.0.0.1:8211", max_
request_duration: "1m30s", max_request_size: "33554432", tls: "enabled")
               Log Level: info
                   Mlock: supported: true, enabled: false
           Recovery Mode: false
                 Storage: file
                 Version: Vault v1.5.4+ent
             Version Sha: 1d81c1e64854fb0dcb3323468d95ad5590460a40

WARNING! dev mode is enabled! In this mode, Vault runs entirely in-memory
and starts unsealed with a single unseal key. The root token is already
authenticated to the CLI, so you can immediately begin using Vault.

You may need to set the following environment variable:

    $ export VAULT_ADDR='http://127.0.0.1:8200'

The unseal key and root token are displayed below in case you want to
seal/unseal the Vault or re-authenticate.

Unseal Key: I29KTEqQVcl2Pa3xKgXffcwP9ae0ow157NFuG7Pj14A=
Root Token: s.XtzYp0lIJtaW3fMAtgWHdXxo

Development mode should NOT be used in production installations!

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

You should see output similar to that above. Notice that **Unseal Key** and **Root Token** values are displayed.

NOTE: The dev server stores all its data in-memory (but still encrypted), listens on **localhost** without TLS, and automatically unseals and shows you the unseal key and root access key.

With the dev server started, perform the following:

1. Launch a new terminal session.
2. Copy and run the export **VAULT_ADDR** ... command from the terminal output. This will configure the Vault client to talk to the dev server.

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

   Vault CLI determines which Vault servers to send requests using the **VAULT_ADDR** environment variable.

3. Save the unseal key somewhere. Don't worry about *how* to save this securely. For now, just save it anywhere.
4. Set the **VAULT_TOKEN** environment variable value to the generated **Root Token** value displayed in the terminal output.

   ```
   $ export VAULT_TOKEN="s.akT1I498dqOy4Z2C5ZimAS1R"
   ```

   To interact with Vault, you must provide a valid token. Setting this environment variable is a way to provide the token to Vault via CLI.

Verify the Server is Running

Verify the server is running by running the vault status command. If it ran successfully, the output should look like the following:

```
$ vault status
Key             Value
---             -----
Seal Type       shamir
Initialized     true
Sealed          false
Total Shares    1
Threshold       1
Version         1.5.4+ent
Cluster Name    vault-cluster-8667d21d
Cluster ID      b3977a72-9be9-d900-c0ec-c6012b1902da
HA Enabled      false
```

IMPORTANT: If using the Enterprise version of Vault, the dev server will seal itself 30 minutes after it is started. This means that it will be necessary to perform the following actions every time that the dev server times out, if using the Enterprise version of Vault:

1. Trigger a Vault shutdown with CTRL+C in the terminal window where the Vault server was running.
2. Run the following command in a terminal:

   ```
   $ rm -r /tmp/vault/
   ```

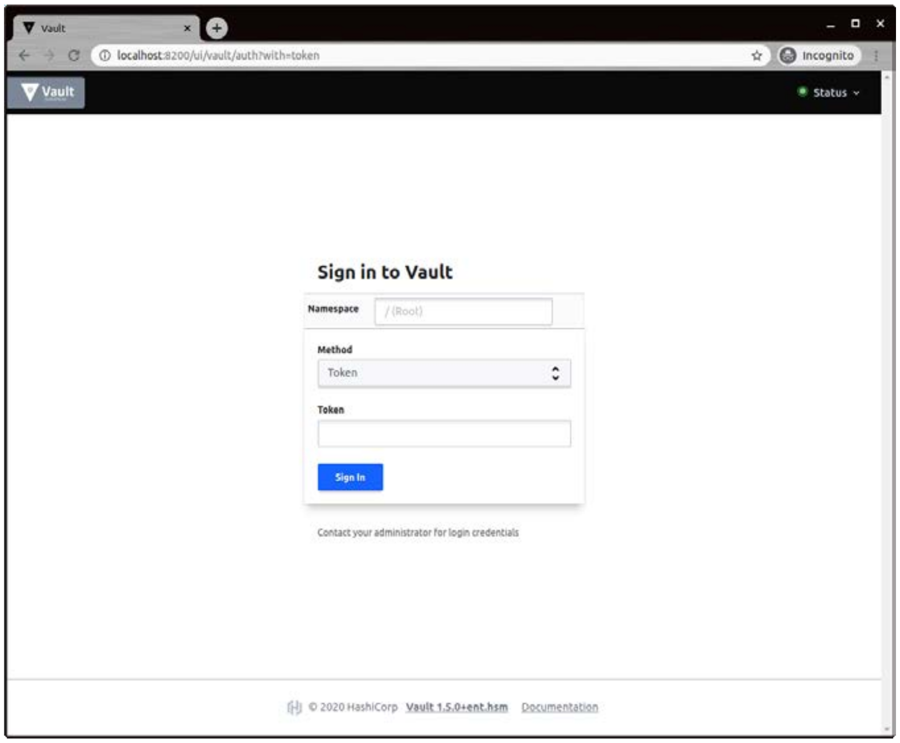
3. Re-perform steps 3.2 thru 3.5.
4. Re-configure either the **userpass** or **TLS authentication** auth method in Vault, as described in section 4.

For more information about how the dev server works, please refer to https://www.vaultproject.io/docs/concepts/dev-server.

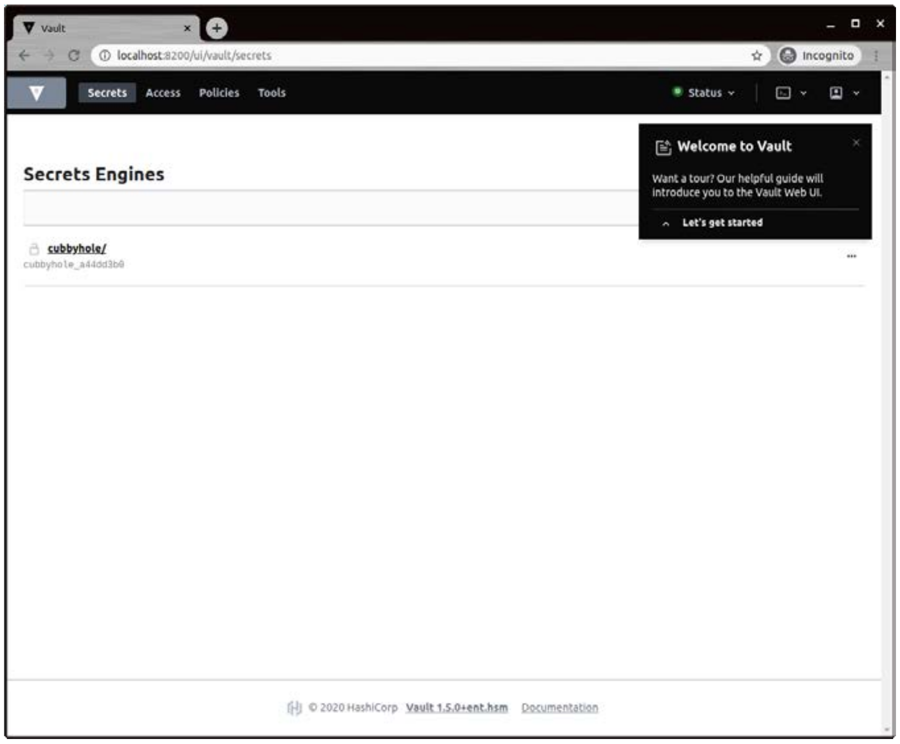
As stated previously, the information provided in this integration guide can be applied to a production implementation of Vault. For specifics on how to deploy Vault in a production environment please refer to HashiCorp's Vault documentation (https://learn.hashicorp.com/tutorials/vault/getting-started-deploy).

[4.4] ACCESSING THE VAULT UI

Go to http://localhost:8200 in a web browser.

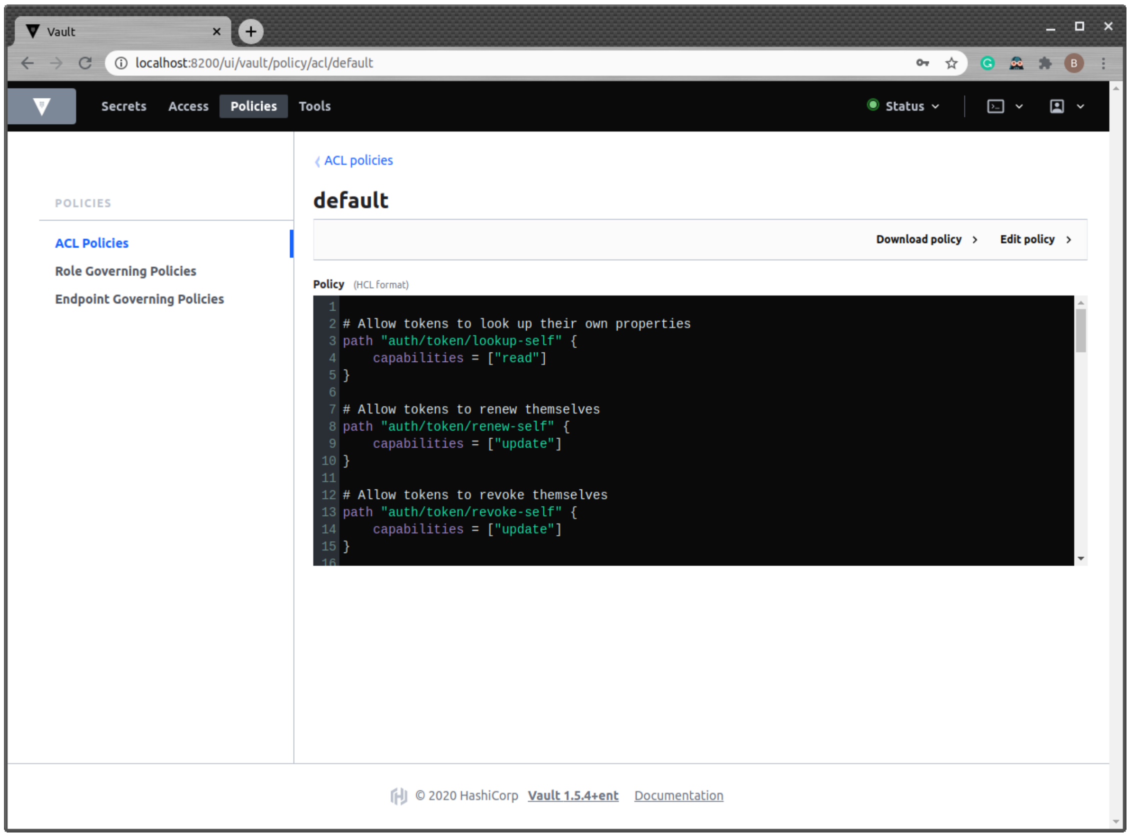

Vault
localhost:8200/ui/vault/auth?with=token
Incognito
Vault
Status
Sign in to Vault
Namespace
/ (Root)
Method
Token
Token
Sign In
Contact your administrator for login credentials
© 2020 HashiCorp Vault 1.5.0+ent.hsm Documentation

Copy and paste in the Root Token that was output from the **vault server** command into the "Token" field, then click "Sign In".


Vault
localhost:8200/ui/vault/secrets
Incognito
Secrets
Access
Policies
Tools
Status
Secrets Engines
Welcome to Vault
Want a tour? Our helpful guide will introduce you to the Vault Web UI.
Let's get started
cubbyhole/
cubbyhole_a44dd3b0
© 2020 HashiCorp Vault 1.5.0+ent.hsm Documentation

[4.5] MODIFY THE DEFAULT ACL POLICY

Navigate to the Policies menu, then select the "default" ACL policy.



```

# Allow tokens to look up their own properties
path "auth/token/lookup-self" {
    capabilities = ["read"]
}

# Allow tokens to renew themselves
path "auth/token/renew-self" {
    capabilities = ["update"]
}

# Allow tokens to revoke themselves
path "auth/token/revoke-self" {
    capabilities = ["update"]
}
```

Click "Edit Policy", then scroll to the bottom of the policy and paste in the following starting at line 89:

```
path "secret/data/*" {
  capabilities = ["create", "read", "update", "delete", "list"]
}
path "secret/*" {
  capabilities = ["create", "read", "update", "delete", "list"]
}
path "sys/*" {
  capabilities = [ "create", "read", "update", "delete", "list" ]
}
path "sys/mounts/*" {
  capabilities = [ "create", "read", "update", "delete", "list" ]
}
# List enabled secrets engine
path "sys/mounts" {
  capabilities = [ "create", "read", "update", "delete", "list" ]
}
# Work with pki secrets engine
path "pki*" {
  capabilities = [ "create", "read", "update", "delete", "list", "sudo" ]
}
```

Click "Save", then a message should appear in the bottom left-hand side of the page confirming that ACL policy "default" was successfully saved.

[5] SETTING UP AUTHENTICATION BETWEEN THE KMES SERIES 3 AND VAULT

Two different methods can be used to authenticate the KMES Series 3 with Vault, which is the **Userpass Auth Method** or the **TLS Certificates Auth Method**. Instructions for both methods are provided in the following sections.

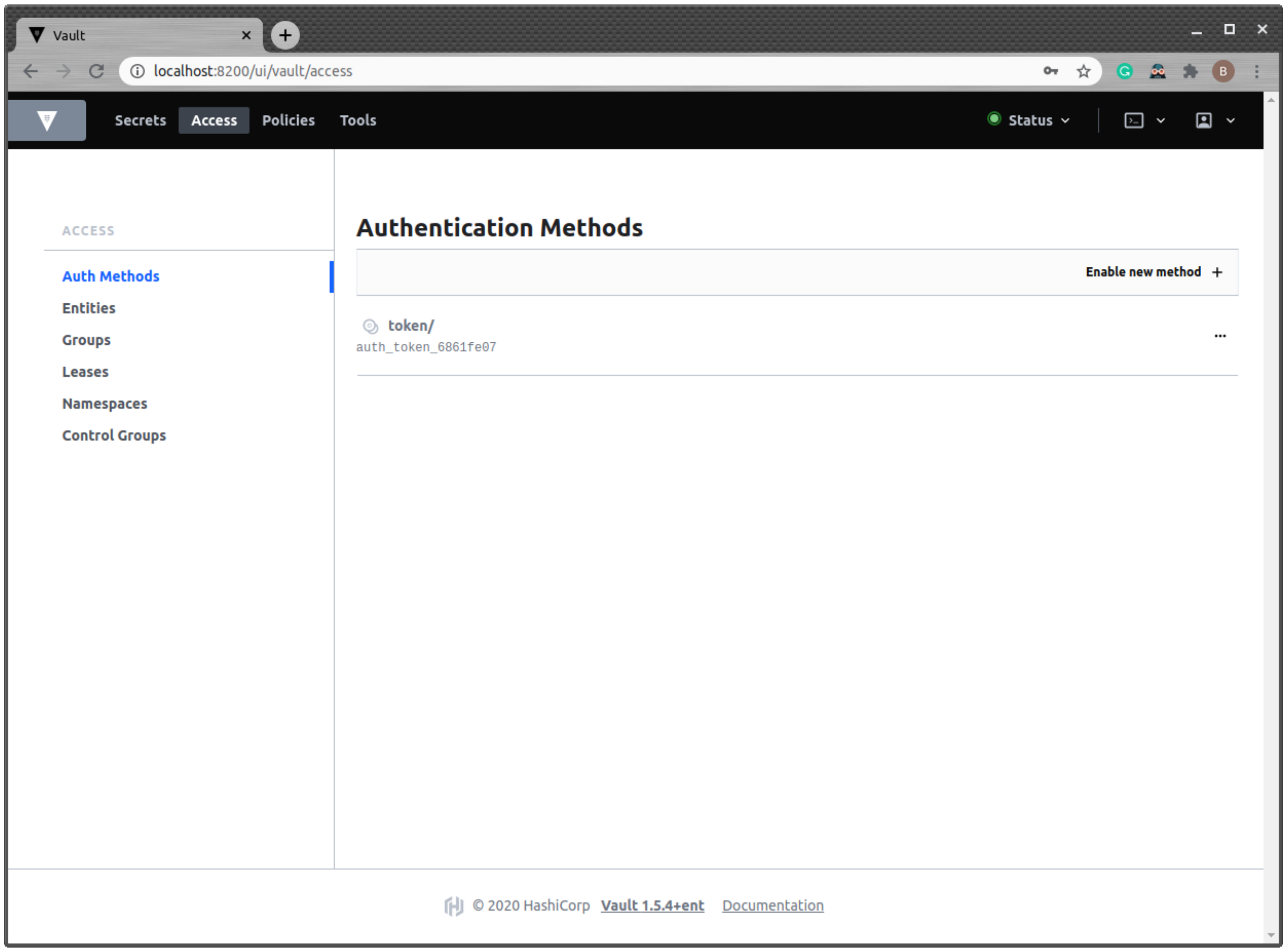
[5.1] USERPASS AUTH METHOD

The **userpass** auth method allows the KMES Series 3 to authenticate with Vault using a username and password combination.

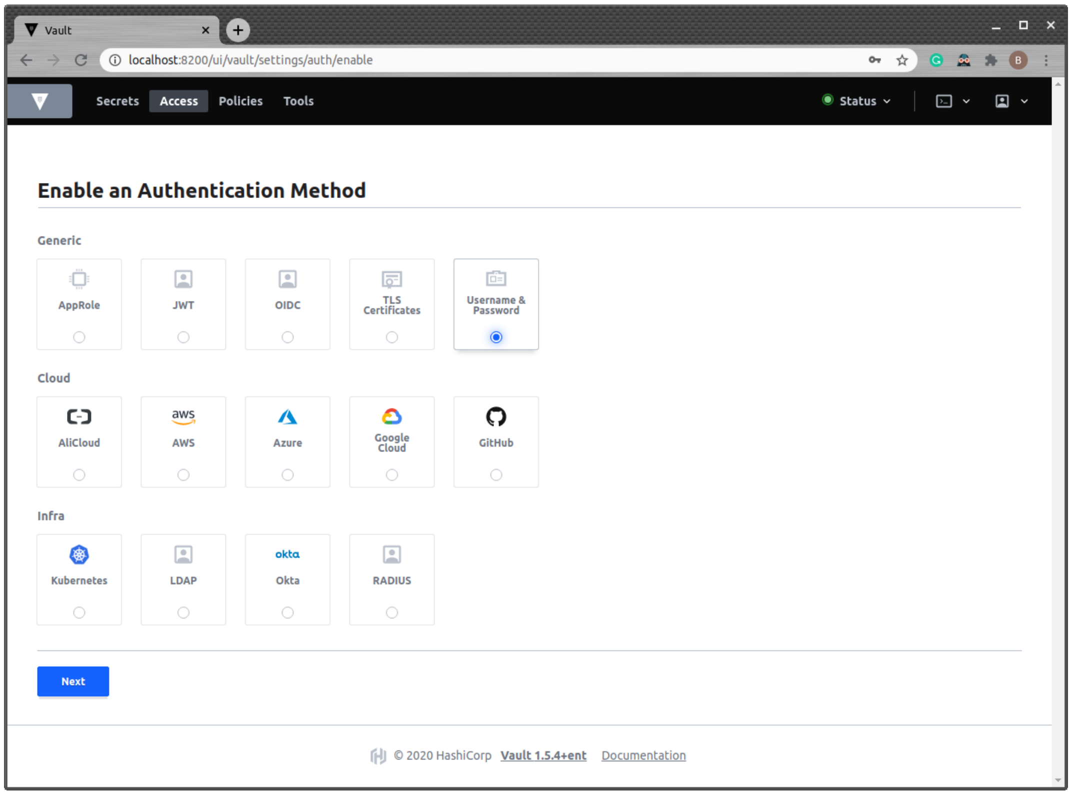
Configuring userpass authentication in Vault

Method 1 - Using the Vault UI

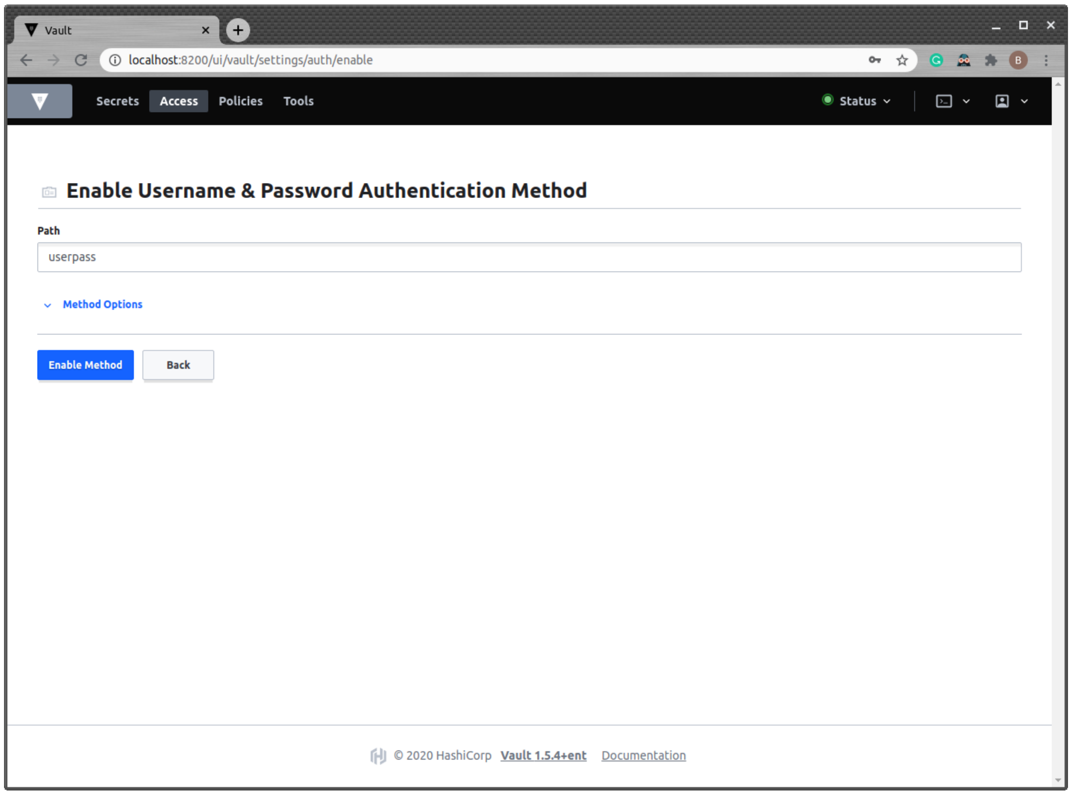
Navigate to the Access page in the Vault UI, then select "Enable new method".



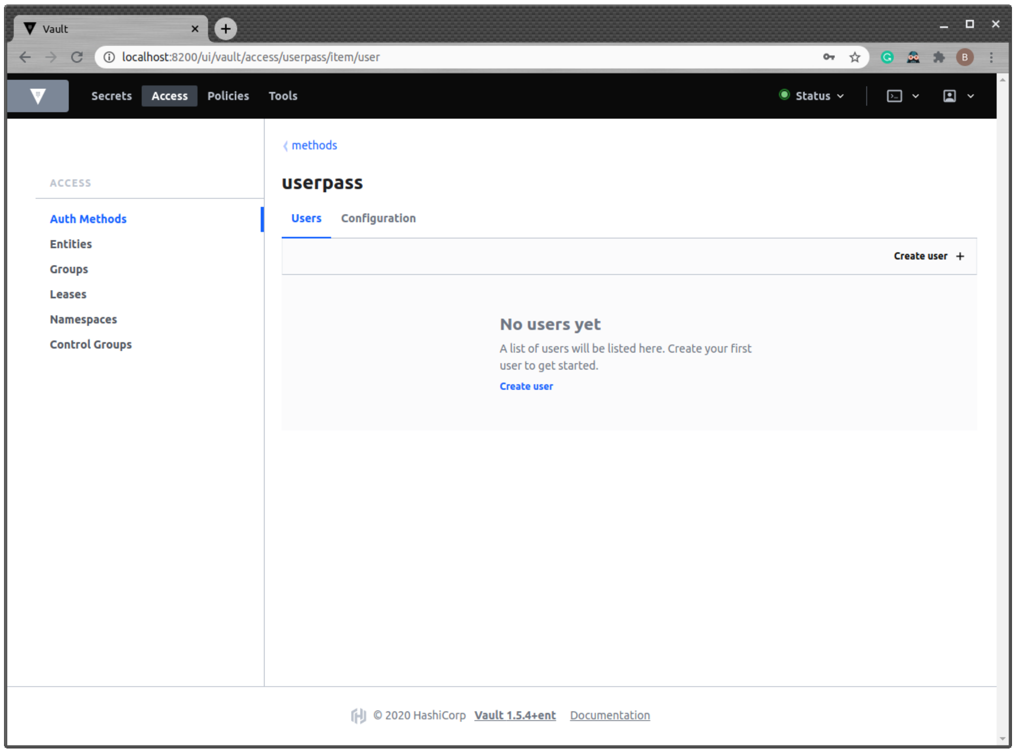
Select the "Username & Password" authentication method, then click "Next".



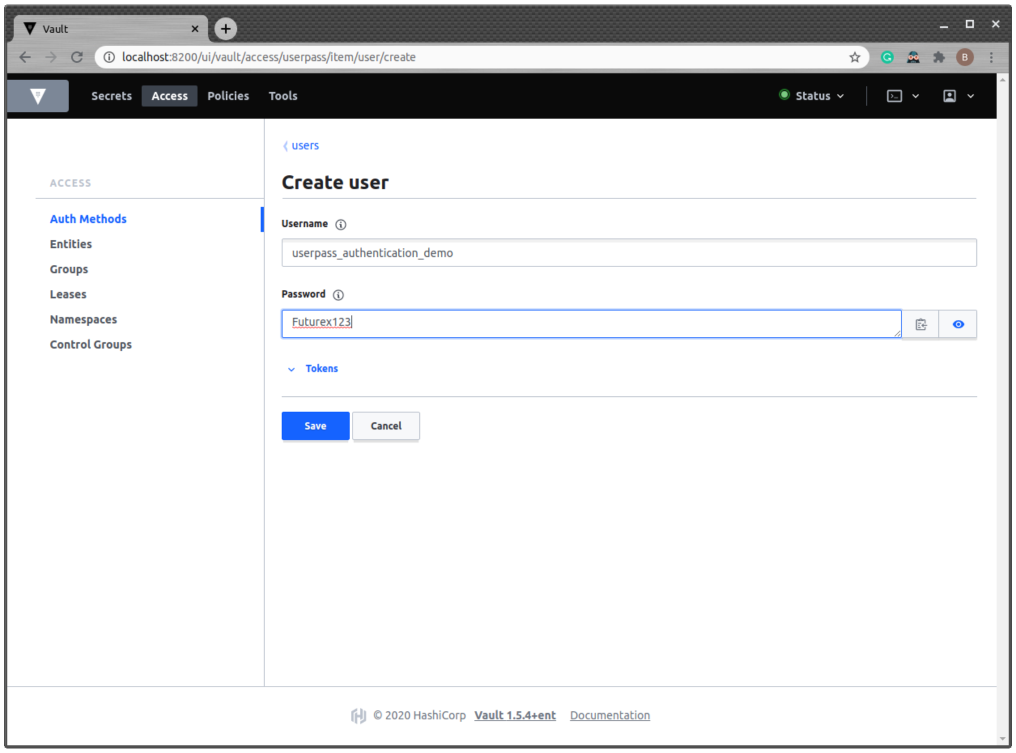
Leave the path as the default value, "userpass", then click "Enable Method".


Vault
localhost:8200/ui/vault/settings/auth/enable
Secrets
Access
Policies
Tools
Status
Enable Username & Password Authentication Method
Path
userpass
Method Options
Enable Method
Back
© 2020 HashiCorp Vault 1.5.4+ent Documentation

Navigate back to the menu for the "userpass" auth method just created, then click "Create user".


Vault
localhost:8200/ui/vault/access/userpass/item/user
Secrets
Access
Policies
Tools
Status
methods
ACCESS
Auth Methods
Entities
Groups
Leases
Namespaces
Control Groups
userpass
Users
Configuration
Create user +
No users yet
A list of users will be listed here. Create your first user to get started.
Create user
© 2020 HashiCorp Vault 1.5.4+ent Documentation

Specify a username and password for the new user, then click "Save".



Username: userpass_authentication_demo

Password: Futurex123

A message should appear in the bottom left-hand side of the page confirming that the new "userpass_ authentication_demo" user was saved successfully.

Method 2 - Using the Vault CLI

Enable the **userpass** auth method:

```
$ vault auth enable userpass

Success! Enabled userpass auth method at: userpass/
```

Configure it with users that are allowed to authenticate:

```
$ vault write auth/userpass/users/userpass_authentication_demo \
    password=Futurex123 \
    policies=admins

Success! Data written to: auth/userpass/users/userpass_authentication_demo
```

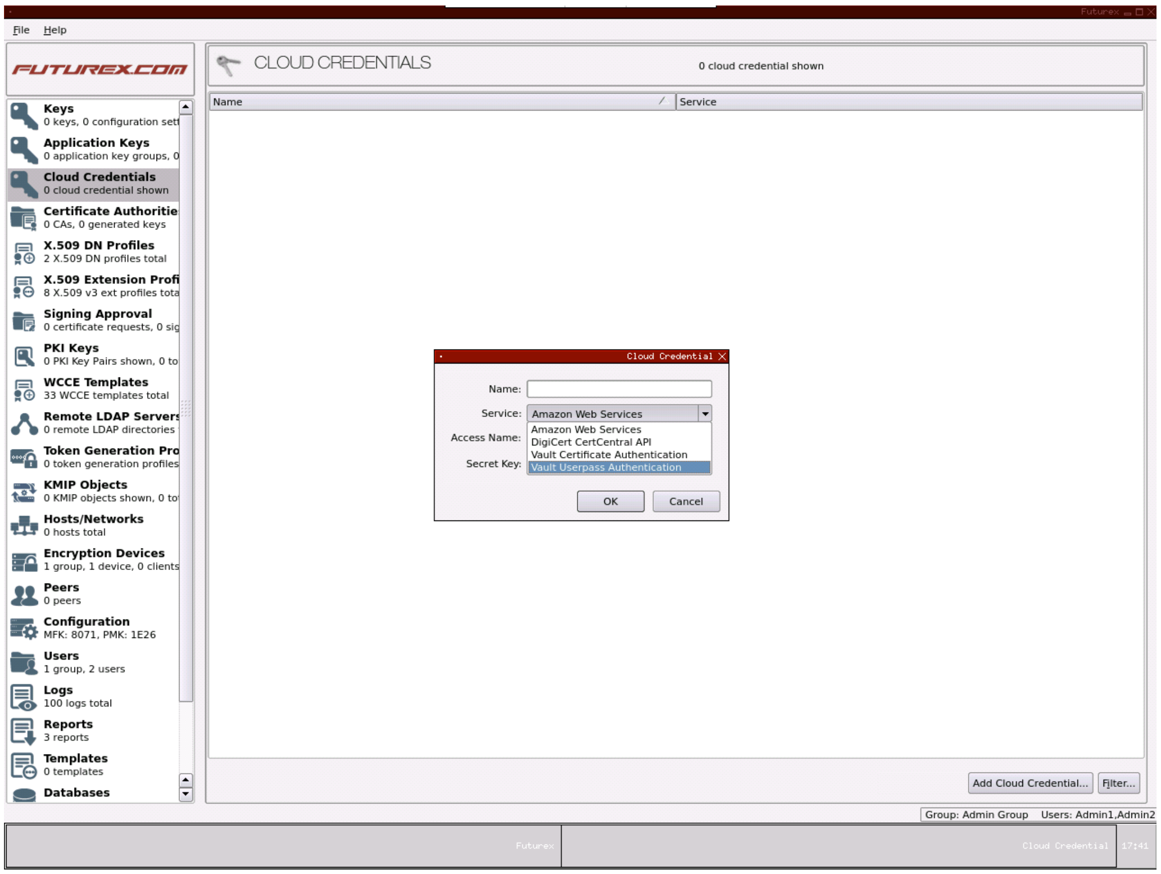
This creates a new user "userpass_authentication_demo" with the password "Futurex123" that will be associated with the "admins" policy. This is the only configuration necessary.

Create a Vault Userpass Authentication Cloud Credential

Log in to the KMES Series 3 application interface with the default Admin users.

Navigate to the *Cloud Credentials* menu, then click the "Add Cloud Credential..." button.

Click the "Service" dropdown and select "Vault Userpass Authentication".

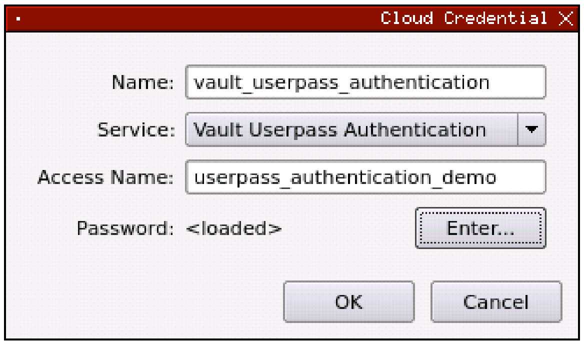


Service: Amazon Web Services
- Amazon Web Services
- DigiCert CertCentral API
- Vault Certificate Authentication
- Vault Userpass Authentication

Any value can be specified in the "Name" field, but the "Access Name" value must match the name of the user that was created under the userpass auth method in Vault.

In the "Password" field, click the "Enter" button and set the same password that was set for the user created in Vault. Click "Save".

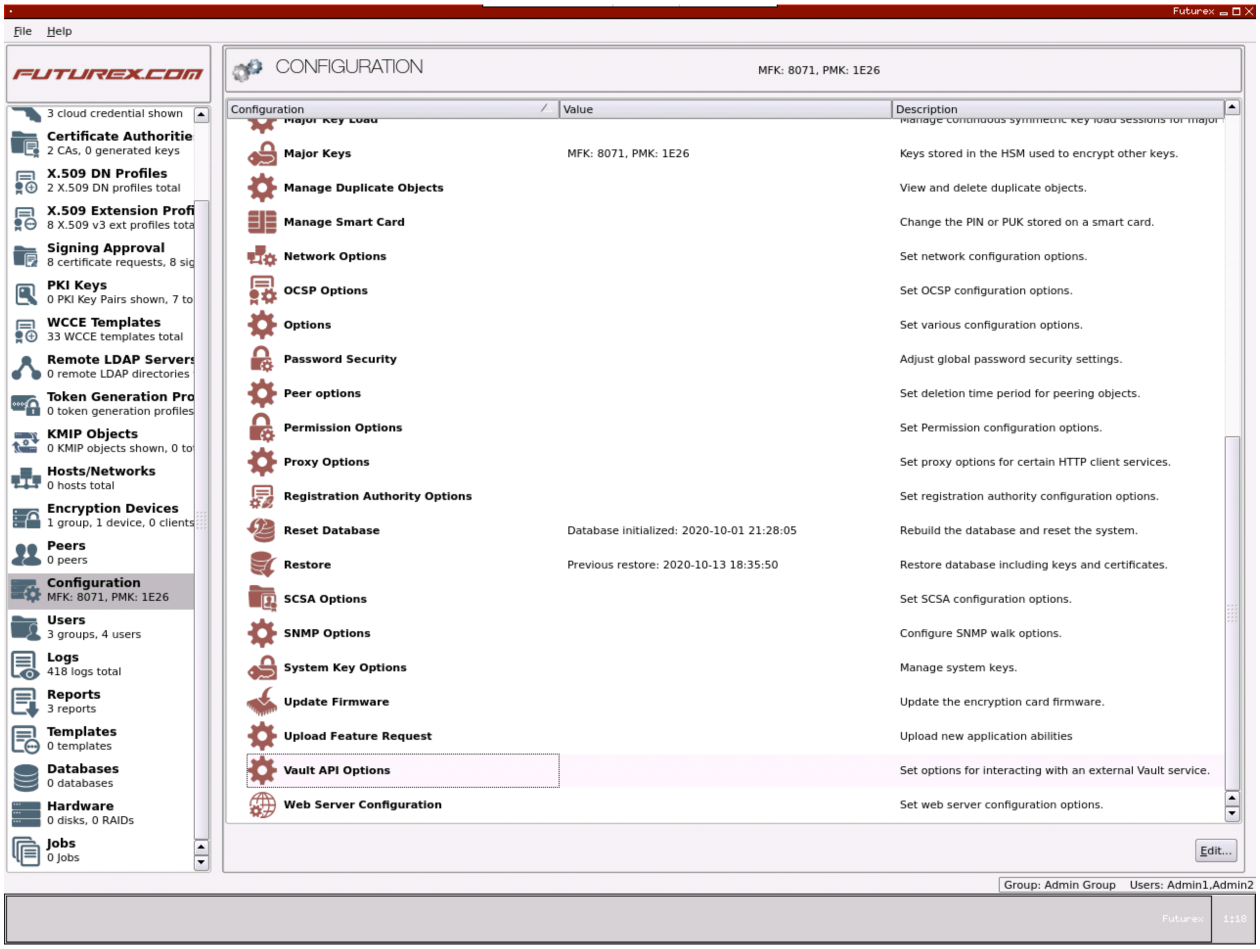
The *Add Cloud Credential* dialog should look similar to the image below, then click "OK".



Name:	vault_userpass_authentication
Service:	Vault Userpass Authentication
Access Name:	userpass_authentication_demo
Password:	<loaded>

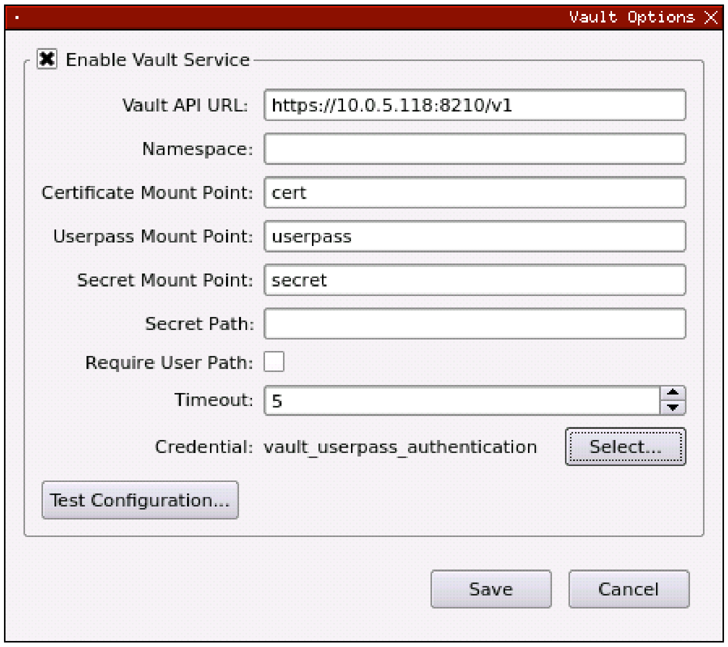
Testing userpass authentication

Navigate to the *Configuration* menu, then double-click on "Vault API Options".



Configuration	Value	Description
Major Key Load		Manage continuous symmetric key load sessions for major
Major Keys	MFK: 8071, PMK: 1E26	Keys stored in the HSM used to encrypt other keys.
Manage Duplicate Objects		View and delete duplicate objects.
Manage Smart Card		Change the PIN or PUK stored on a smart card.
Network Options		Set network configuration options.
OCSP Options		Set OCSP configuration options.
Options		Set various configuration options.
Password Security		Adjust global password security settings.
Peer options		Set deletion time period for peering objects.
Permission Options		Set Permission configuration options.
Proxy Options		Set proxy options for certain HTTP client services.
Registration Authority Options		Set registration authority configuration options.
Reset Database	Database initialized: 2020-10-01 21:28:05	Rebuild the database and reset the system.
Restore	Previous restore: 2020-10-13 18:35:50	Restore database including keys and certificates.
SCSA Options		Set SCSA configuration options.
SNMP Options		Configure SNMP walk options.
System Key Options		Manage system keys.
Update Firmware		Update the encryption card firmware.
Upload Feature Request		Upload new application abilities
Vault API Options		Set options for interacting with an external Vault service.
Web Server Configuration		Set web server configuration options.

Check the "Enable Vault Service" box.



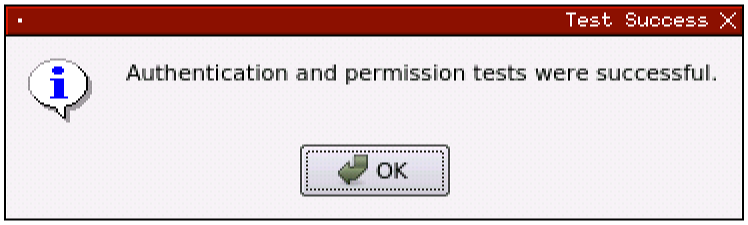
Vault Options

☒ Enable Vault Service

Vault API URL:	https://10.0.5.118:8210/v1
Namespace:	
Certificate Mount Point:	cert
Userpass Mount Point:	userpass
Secret Mount Point:	secret
Secret Path:	
Require User Path:	☐
Timeout:	5
Credential:	vault_userpass_authentication

Set the Vault API URL to "https://10.0.5.118:8210/v1" and select the Vault Userpass Authentication Cloud Credential created in the previous step.

The rest of the fields can be left as their default values. Click "Test Configuration...".



Test Success

Authentication and permission tests were successful.

If all of the configuration steps were completed properly the test should result in a success.

[5.2] TLS CERTIFICATES AUTH METHOD

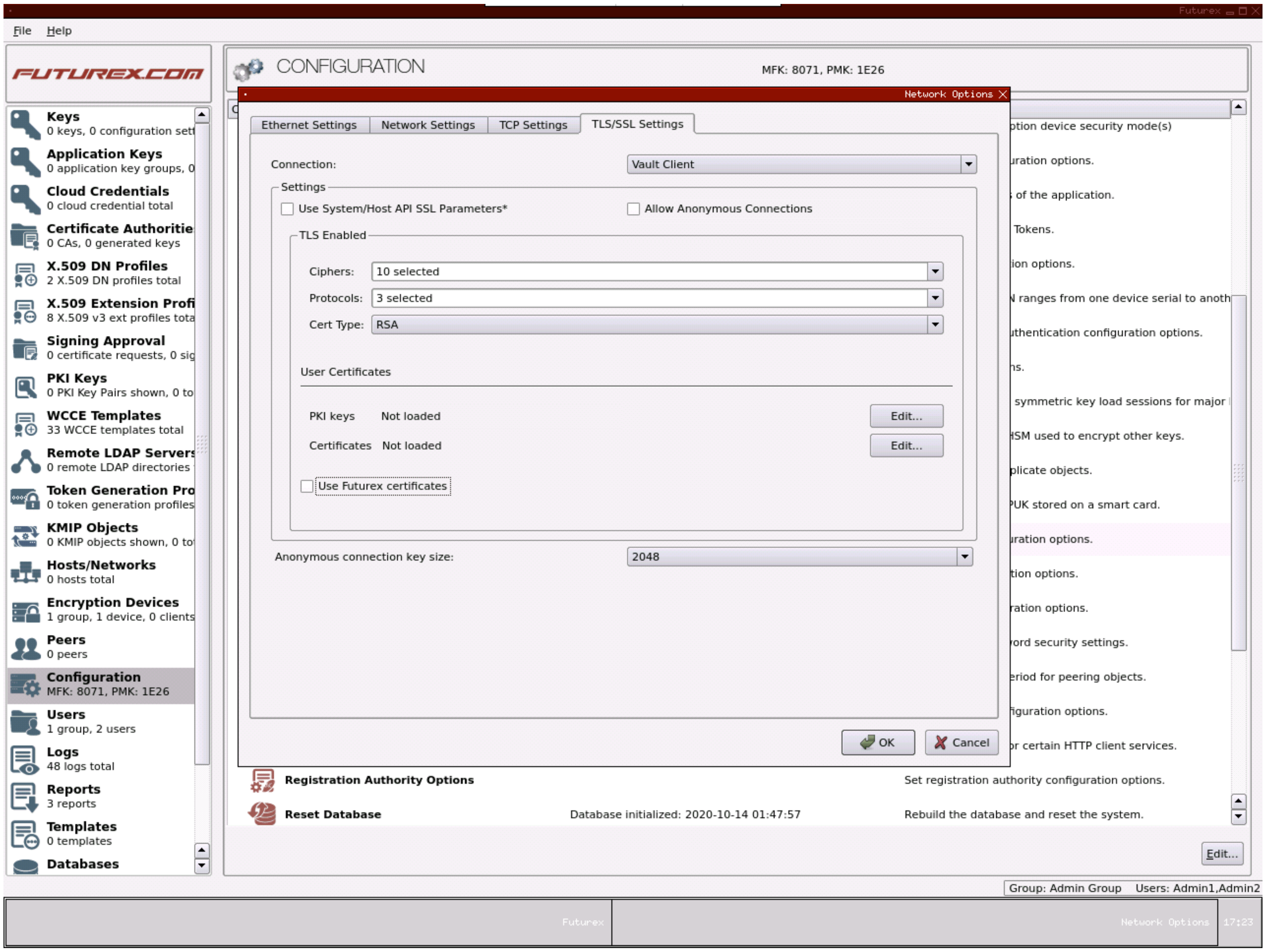
The **cert** auth method allows the KMES Series 3 to authenticate with Vault using SSL/TLS client certificates which are either signed by a CA or self-signed.

Configure the Vault Client connection pair on the KMES Series 3

For the TLS Certificates auth method, it is necessary to first configure the **Vault Client** connection pair on the KMES Series 3 before configuring the **cert** authentication method in Vault.

Log in to the KMES Series 3 application interface with the default Admin users.

Navigate to the *Configuration* menu, then double-click on *Network Options*. Go to the *TLS/SSL Settings* tab, then click the dropdown and switch to the **Vault Client** connection pair.



MFK: 8071, PMK: 1E26

Keys
0 keys, 0 configuration set
Application Keys
0 application key groups, 0
Cloud Credentials
0 cloud credential total
Certificate Authoritie
0 CAs, 0 generated keys
X.509 DN Profiles
2 X.509 DN profiles total
X.509 Extension Profi
8 X.509 v3 ext profiles tota
Signing Approval
0 certificate requests, 0 sig
PKI Keys
0 PKI Key Pairs shown, 0 to
WCCE Templates
33 WCCE templates total
Remote LDAP Servers
0 remote LDAP directories
Token Generation Pro
0 token generation profiles
KMIP Objects
0 KMIP objects shown, 0 to
Hosts/Networks
0 hosts total
Encryption Devices
1 group, 1 device, 0 clients
Peers
0 peers
Configuration
MFK: 8071, PMK: 1E26
Users
1 group, 2 users
Logs
48 logs total
Reports
3 reports
Templates
0 templates
Databases

Connection: Vault Client

Settings

- [ ] Use System/Host API SSL Parameters*
- [ ] Allow Anonymous Connections

TLS Enabled

Ciphers: 10 selected
Protocols: 3 selected
Cert Type: RSA

User Certificates

PKI keys Not loaded
Certificates Not loaded

- [ ] Use Futurex certificates

Anonymous connection key size: 2048

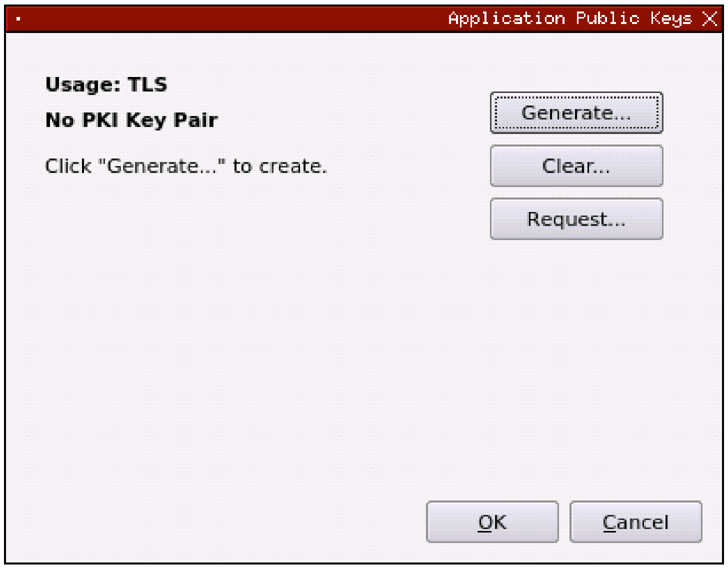
ption device security mode(s)
uration options.
of the application.
Tokens.
ion options.
N ranges from one device serial to anoth
uthentication configuration options.
ns.
symmetric key load sessions for major
HSM used to encrypt other keys.
plicate objects.
PUK stored on a smart card.
uration options.
tion options.
ration options.
ord security settings.
eriod for peering objects.
figuration options.
or certain HTTP client services.

Registration Authority Options — Set registration authority configuration options.
Reset Database — Database initialized: 2020-10-14 01:47:57 — Rebuild the database and reset the system.

Group: Admin Group Users: Admin1,Admin2

Futurex Network Options 17:23

Uncheck the "Use System/Host API SSL Parameters" and "Use Futurex Certificates" boxes, then click the "Edit..." button next to "PKI keys" in the *User Certificates* section.


Application Public Keys

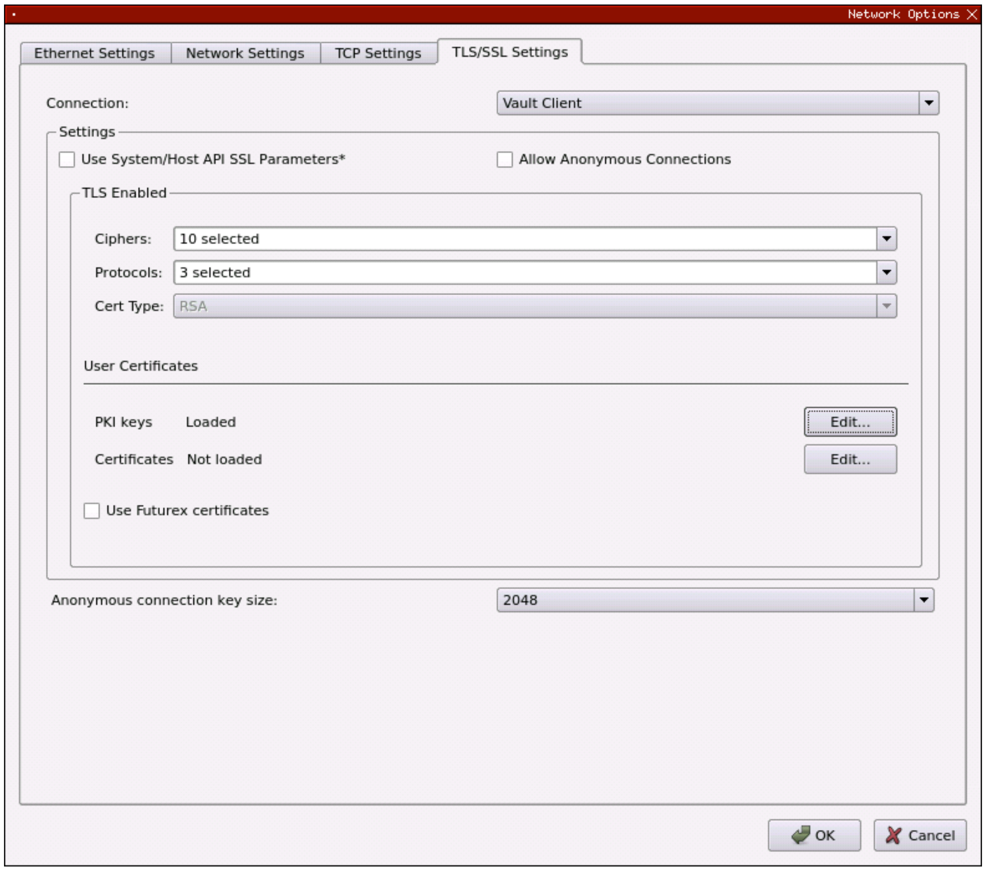
Usage: TLS

No PKI Key Pair

Click "Generate..." to create.

Generate a new PKI key pair, then request a CSR.

Next, the CSR just generated needs to be signed by a Certificate Authority (CA). For this demonstration, the CSR will be signed by the same CA that signed the client certificate that was configured in the Vault configuration file in section 3.2. Upload the CA certificate and the signed CSR to the storage medium configured on the KMES Series 3, then click the "Edit..." button next to "Certificates" in the User Certificates section, as shown below.


Network Options

Connection: Vault Client

Settings

Use System/Host API SSL Parameters* ☐ Allow Anonymous Connections ☐

TLS Enabled

Ciphers: 10 selected

Protocols: 3 selected

Cert Type: RSA

User Certificates

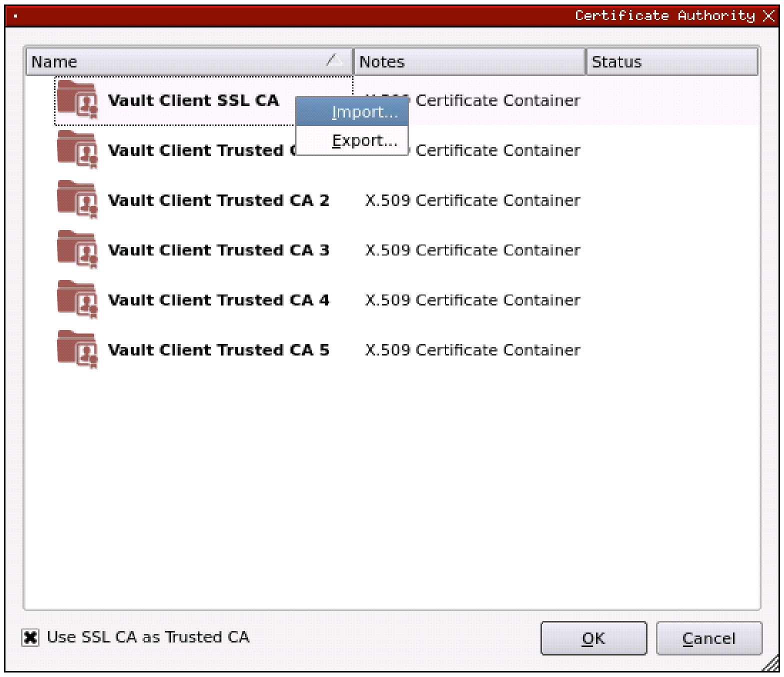
PKI keys Loaded

Certificates Not loaded

☐ Use Futurex certificates

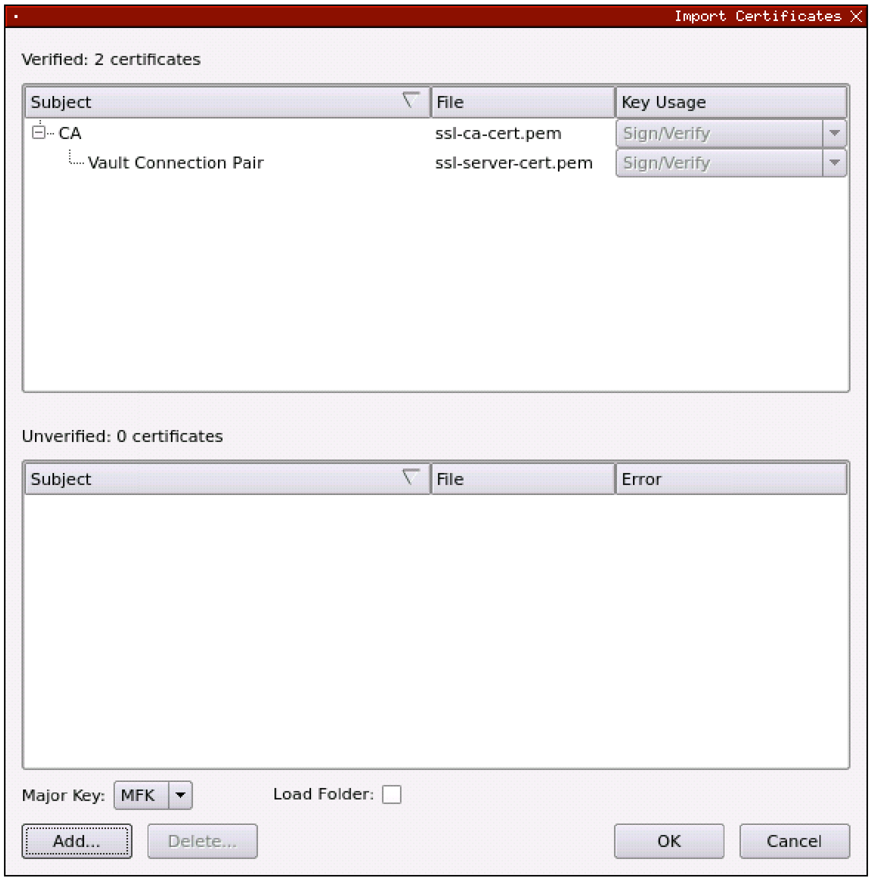
Anonymous connection key size: 2048

Right-click on the "Vault Client SSL CA" X.509 Certificate Container, then click "Import...".



Name	Notes	Status
Vault Client SSL CA	X.509 Certificate Container	
Vault Client Trusted CA 1	X.509 Certificate Container	
Vault Client Trusted CA 2	X.509 Certificate Container	
Vault Client Trusted CA 3	X.509 Certificate Container	
Vault Client Trusted CA 4	X.509 Certificate Container	
Vault Client Trusted CA 5	X.509 Certificate Container	

Click the "Add" button in the lower-left area of the *Import Certificates* dialog, then find, select, and open the CA and signed CSR, and click "OK".



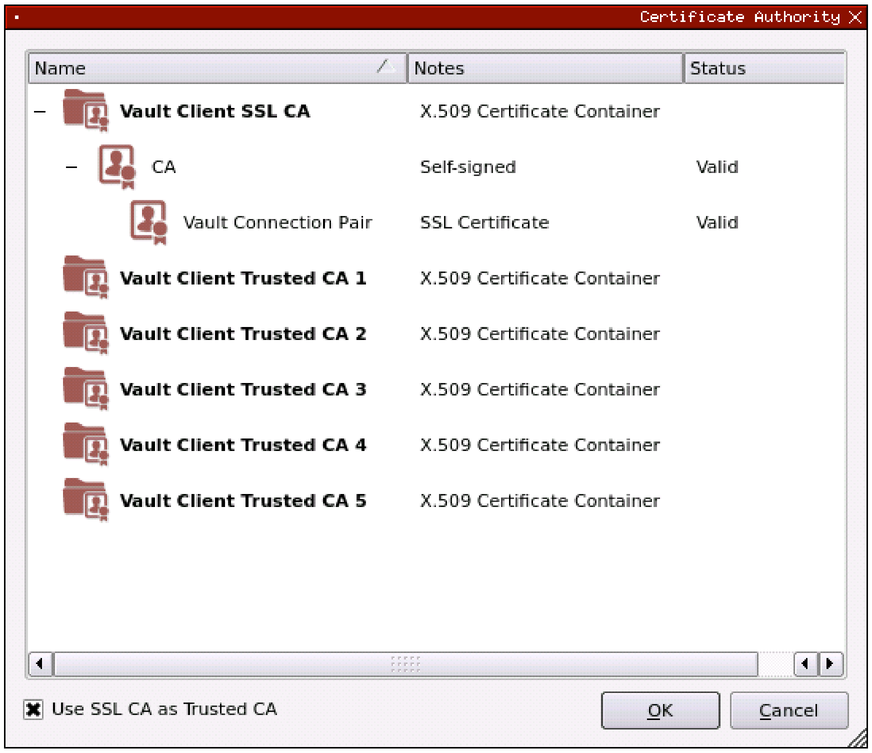
Verified: 2 certificates

Subject	File	Key Usage
CA	ssl-ca-cert.pem	Sign/Verify
Vault Connection Pair	ssl-server-cert.pem	Sign/Verify

Unverified: 0 certificates

Subject	File	Error

The certificate tree for the Vault Client connection pair is loaded now, as shown below.



Certificate Authority

Name	Notes	Status
– **Vault Client SSL CA**	X.509 Certificate Container	
– CA	Self-signed	Valid
Vault Connection Pair	SSL Certificate	Valid
Vault Client Trusted CA 1	X.509 Certificate Container	
Vault Client Trusted CA 2	X.509 Certificate Container	
Vault Client Trusted CA 3	X.509 Certificate Container	
Vault Client Trusted CA 4	X.509 Certificate Container	
Vault Client Trusted CA 5	X.509 Certificate Container	

[✓] Use SSL CA as Trusted CA

Click the "OK" button to save.

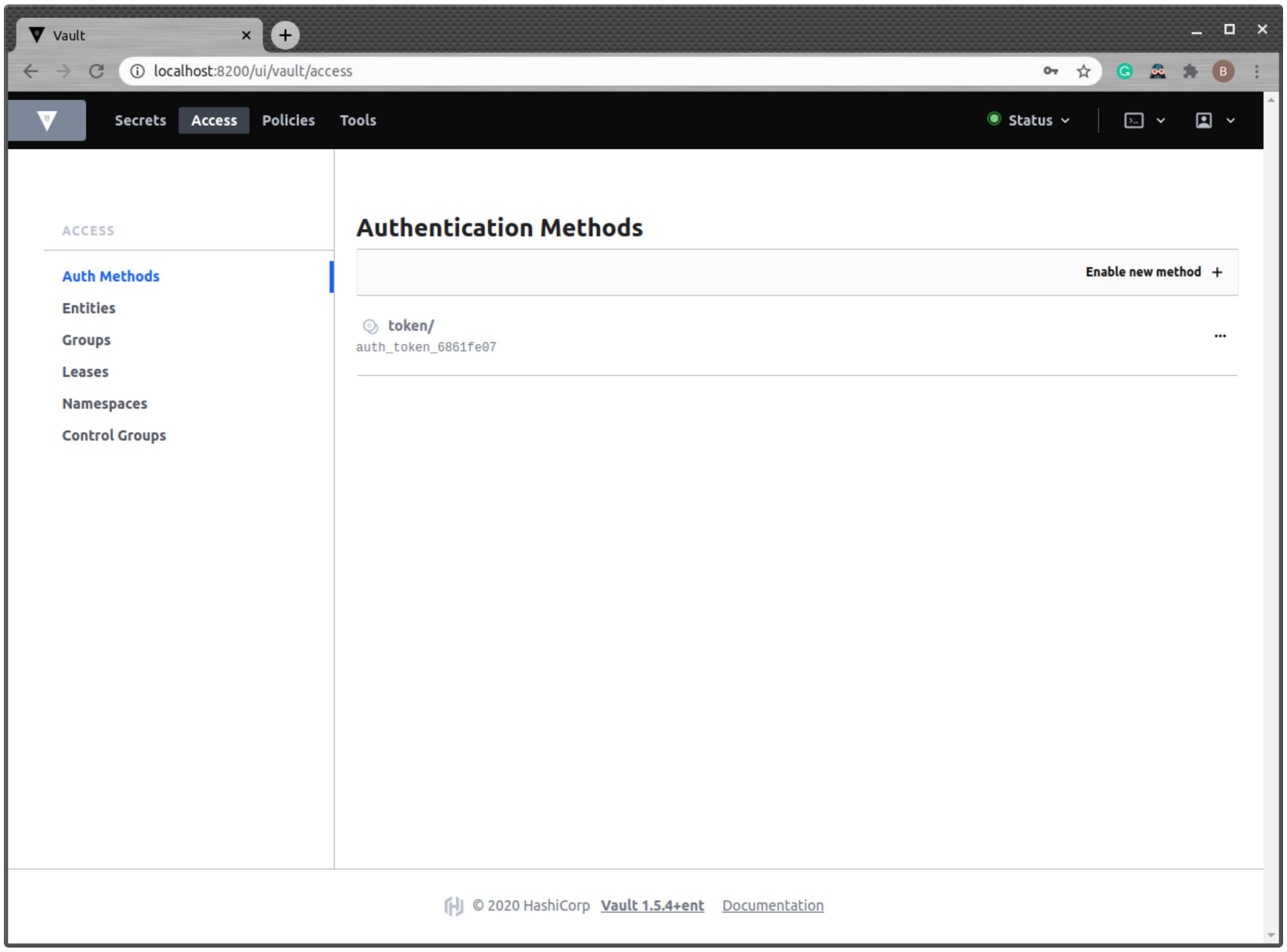
NOTE: As previously noted, for this demonstration, the same CSR was used to sign both the Vault Client connection pair CSR and the client certificate set in the Vault configuration file. If this were not the case, then the CA that signed the Vault Client connection pair CSR would need to be loaded to the "Vault Client SSL CA" X.509 Certificate Container with the accompanying signed certificate, and the CA that signed the client certificate set in the Vault configuration file would need to be loaded to any of the "Vault Client Trusted CA" X.509 Certificate Containers, along with the accompanying signed certificate.

Click the "OK" button to save and exit out of the *Network Options* menu.

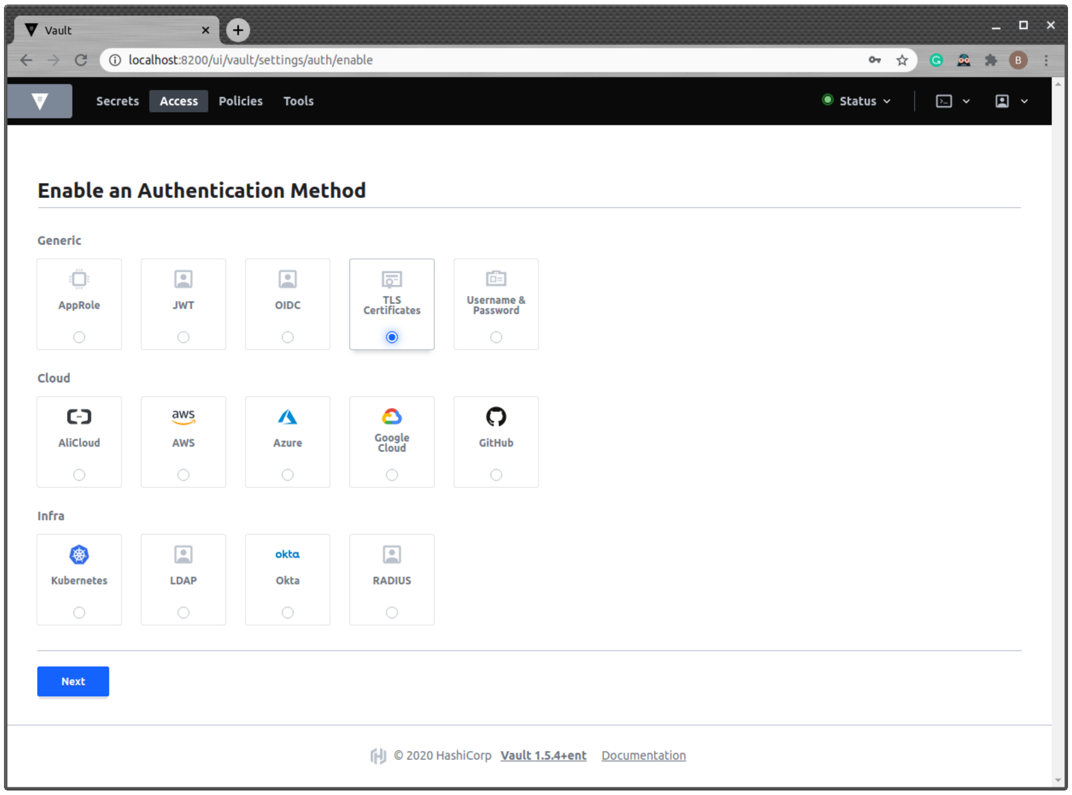
Configuring cert authentication in Vault

Method 1 - Using the Vault UI

Navigate to the Access page in the Vault UI, then select "Enable new method".



Select the "TLS Certificates" authentication method, then click "Next".


Vault
localhost:8200/ui/vault/settings/auth/enable
Secrets
Access
Policies
Tools
Status

Enable an Authentication Method

Generic

AppRole ○ JWT ○ OIDC ○ TLS Certificates ● Username & Password ○

Cloud

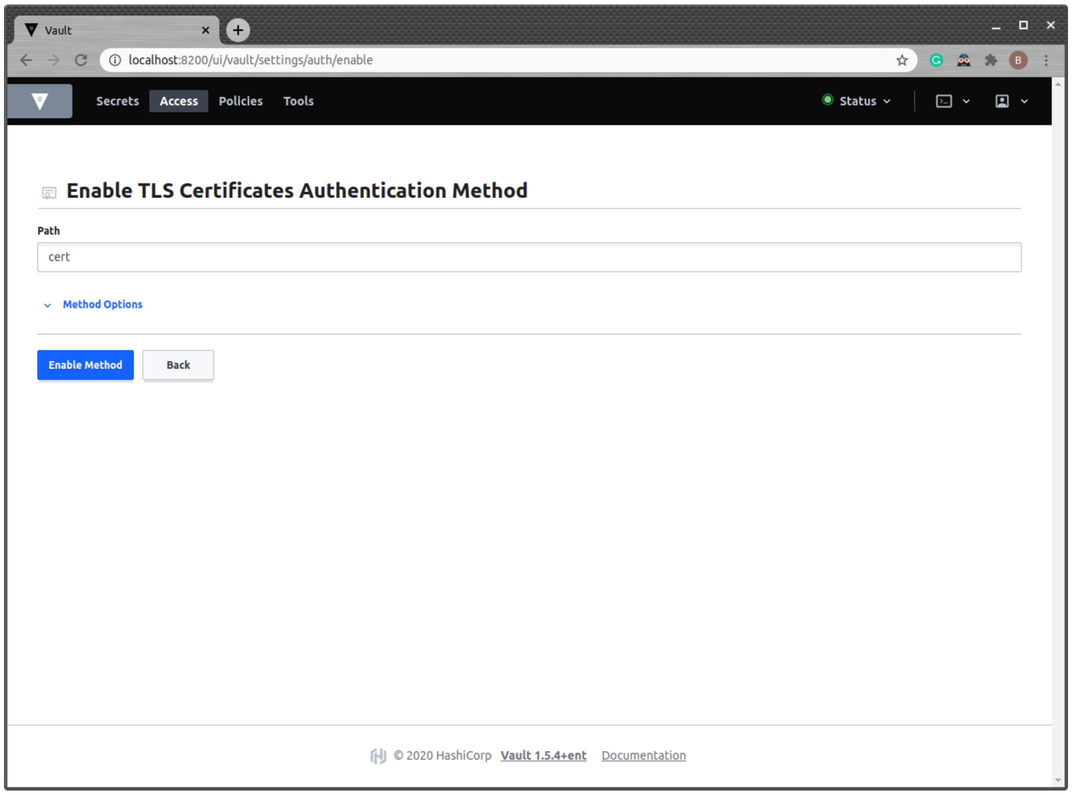
AliCloud ○ AWS ○ Azure ○ Google Cloud ○ GitHub ○

Infra

Kubernetes ○ LDAP ○ Okta ○ RADIUS ○

Next
© 2020 HashiCorp Vault 1.5.4+ent Documentation

Leave the path as the default value, "cert", then click "Enable Method".

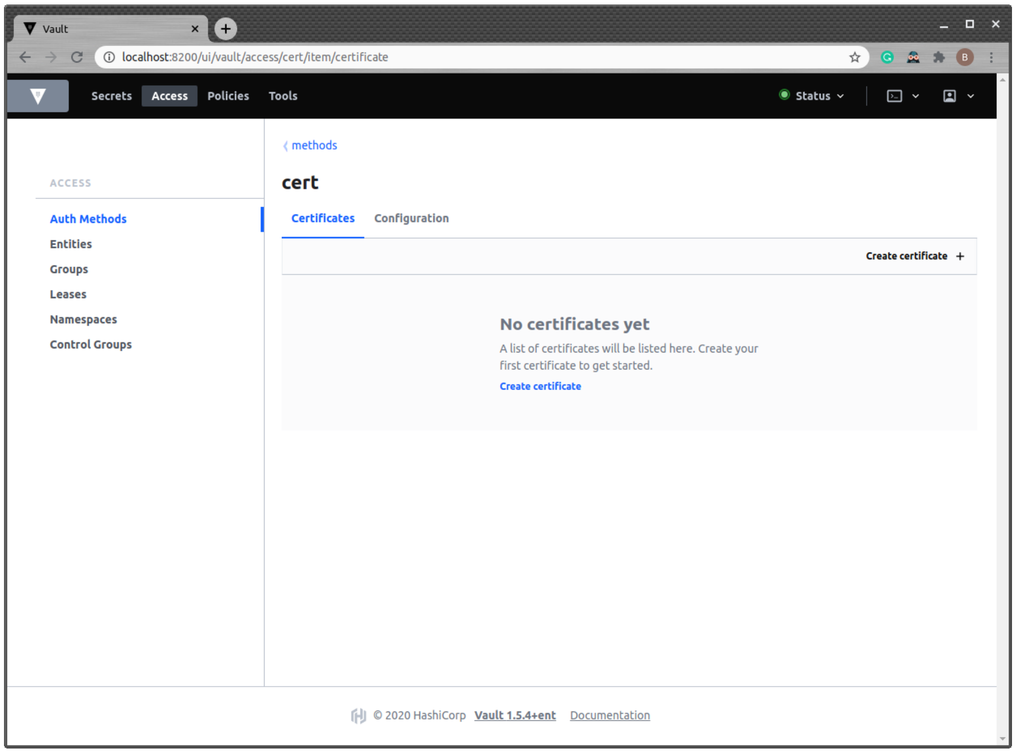

Vault
localhost:8200/ui/vault/settings/auth/enable
Secrets
Access
Policies
Tools
Status

Enable TLS Certificates Authentication Method

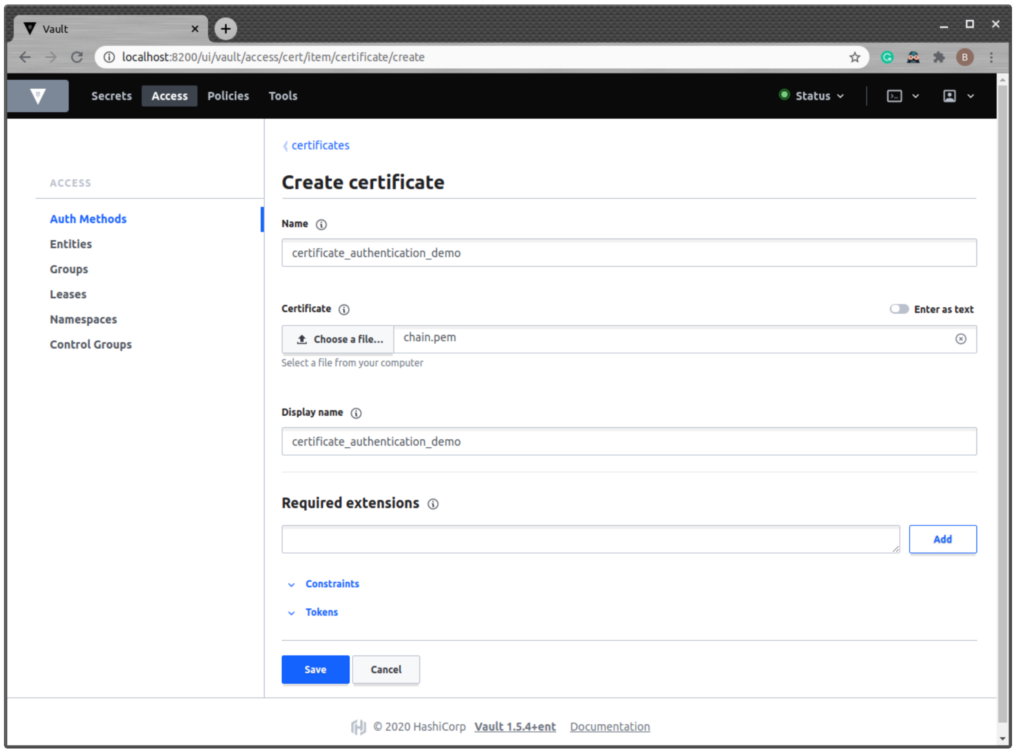
Path: cert

Method Options
Enable Method
Back
© 2020 HashiCorp Vault 1.5.4+ent Documentation

Navigate back to the menu for the "cert" auth method just created, then click "Create certificate".



Specify a name for the certificate and upload a single *.pem* file that contains the certificate chain configured for the Vault Client connection pair on the KMES Series 3. Then click "Save".



A message should appear in the bottom left-hand side of the page confirming that the new "certificate_ authentication_demo" certificate was saved successfully.

Method 2 - Using the Vault CLI

Enable the **cert** auth method:

```
$ vault auth enable cert

Success! Enabled cert auth method at: cert/
```

Configure it with trusted certificates that are allowed to authenticate:

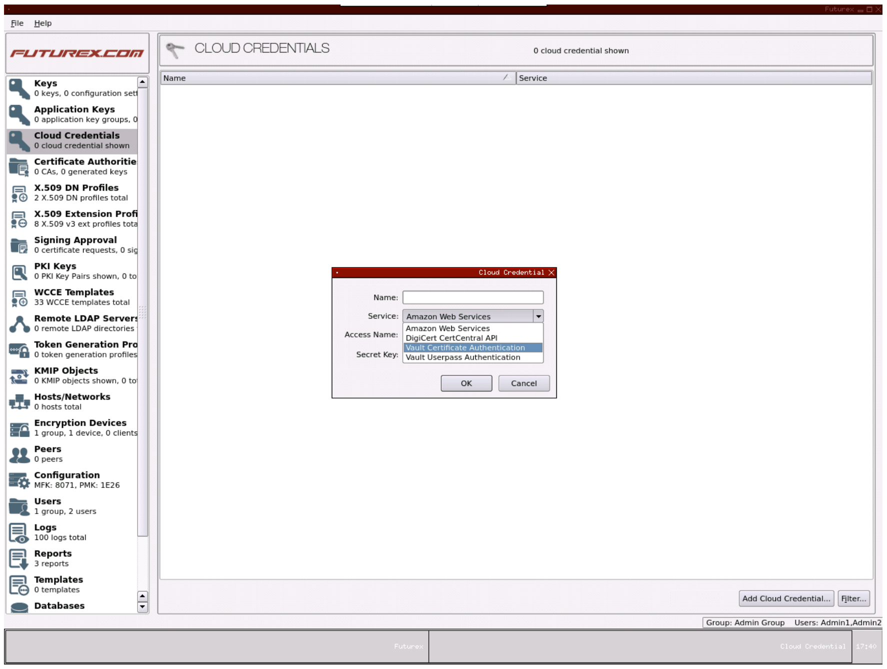
```
$ vault write auth/cert/certs/certificate_authentication_demo \
    display_name=certificate_authentication_demo \
    policies=web,prod \
    certificate=@chain.pem \
    ttl=3600

Success! Data written to: auth/cert/certs/certificate_authentication_demo
```

This creates a new trusted certificate "certificate_authentication_demo" with same display name and the "web" and "prod" policies. The certificate (public key) used to verify clients is given by the "chain.pem" file. Lastly, an optional ttl value can be provided in seconds to limit the lease duration.

Create a Vault Certificate Authentication Cloud Credential

In the KMES Series 3 application interface, navigate to the *Cloud Credentials* menu, then click the "Add Cloud Credential..." button.

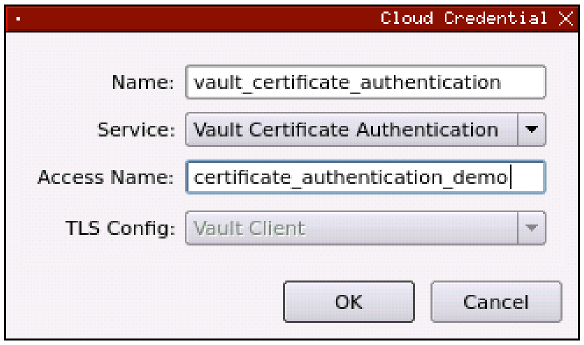
Click the "Service" dropdown and select "Vault Certificate Authentication".



Any value can be specified in the "Name" field, but the "Access Name" value must match the name of the certificate that was created under the cert auth method in Vault.

The value for the "TLS Config" field defaults to "Vault Client". This configures the Cloud Credential to use the "Vault Client" connection pair for authenticating with Vault.

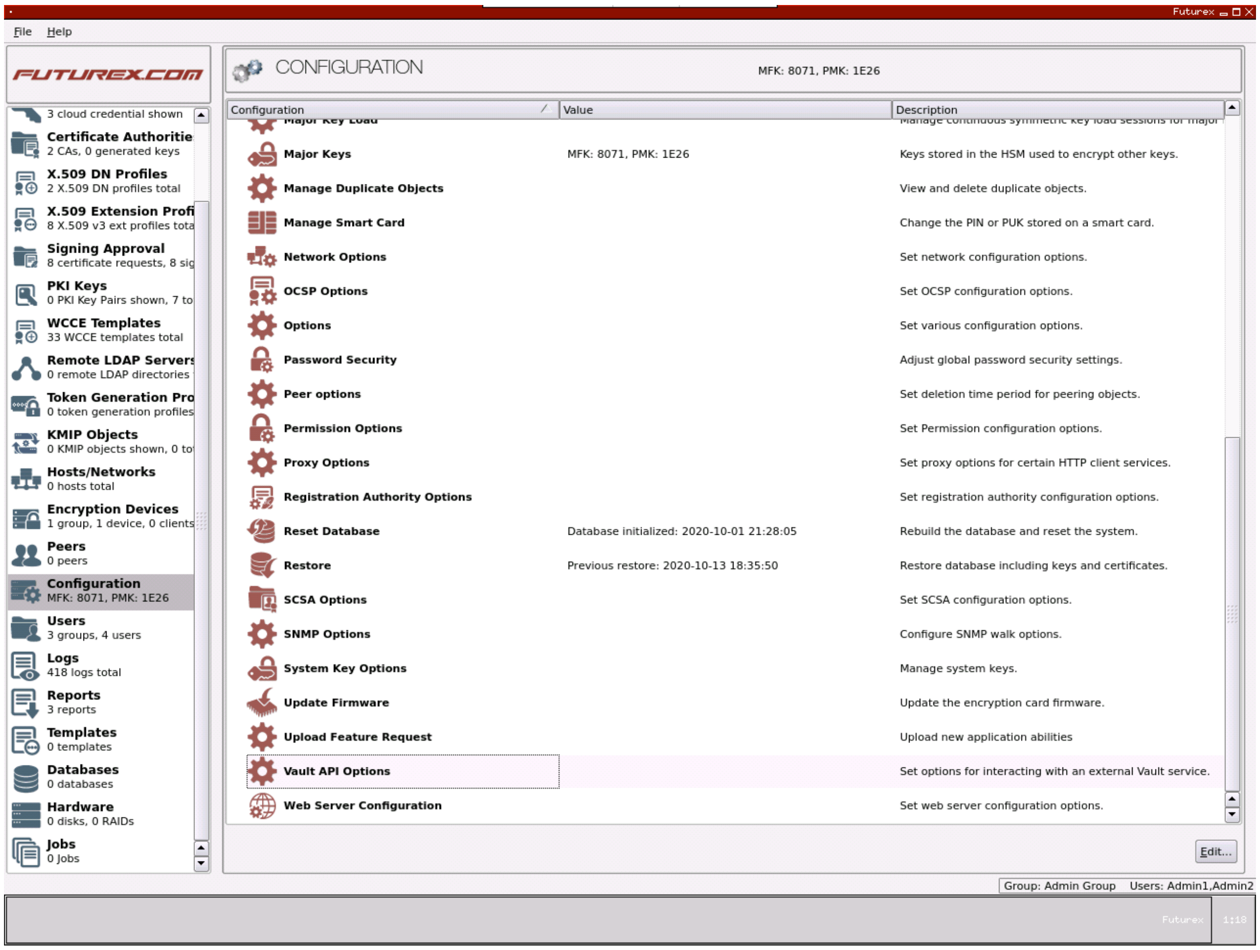
The *Add Cloud Credential* dialog should look similar to the image below, then click "OK".



Name:	vault_certificate_authentication
Service:	Vault Certificate Authentication
Access Name:	certificate_authentication_demo
TLS Config:	Vault Client

Testing cert authentication

Navigate to the *Configuration* menu, then double-click on "Vault API Options".



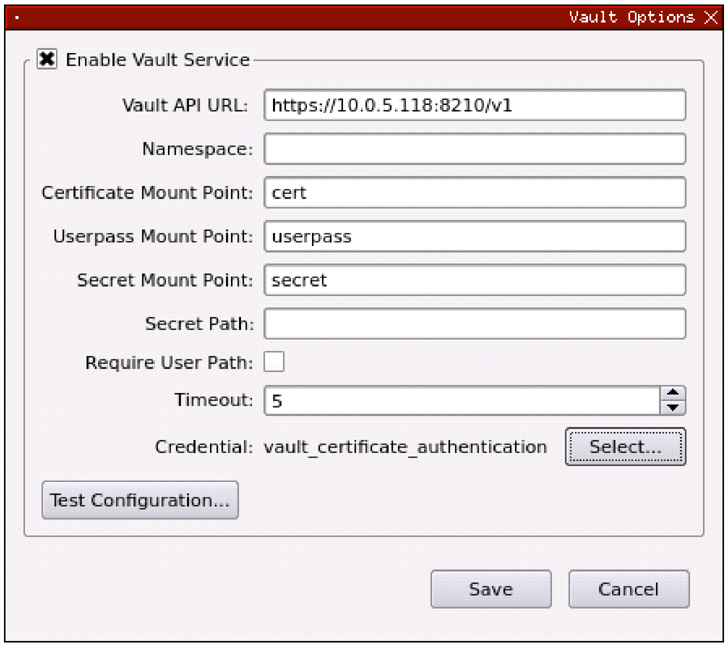
MFK: 8071, PMK: 1E26

3 cloud credential shown
Certificate Authoritie
2 CAs, 0 generated keys
X.509 DN Profiles
2 X.509 DN profiles total
X.509 Extension Profi
8 X.509 v3 ext profiles tota
Signing Approval
8 certificate requests, 8 sig
PKI Keys
0 PKI Key Pairs shown, 7 to
WCCE Templates
33 WCCE templates total
Remote LDAP Server
0 remote LDAP directories
Token Generation Pro
0 token generation profiles
KMIP Objects
0 KMIP objects shown, 0 to
Hosts/Networks
0 hosts total
Encryption Devices
1 group, 1 device, 0 clients
Peers
0 peers
Configuration
MFK: 8071, PMK: 1E26
Users
3 groups, 4 users
Logs
418 logs total
Reports
3 reports
Templates
0 templates
Databases
0 databases
Hardware
0 disks, 0 RAIDs
Jobs
0 Jobs

Configuration	Value	Description
Major Key Load		Manage continuous symmetric key load sessions for major
Major Keys	MFK: 8071, PMK: 1E26	Keys stored in the HSM used to encrypt other keys.
Manage Duplicate Objects		View and delete duplicate objects.
Manage Smart Card		Change the PIN or PUK stored on a smart card.
Network Options		Set network configuration options.
OCSP Options		Set OCSP configuration options.
Options		Set various configuration options.
Password Security		Adjust global password security settings.
Peer options		Set deletion time period for peering objects.
Permission Options		Set Permission configuration options.
Proxy Options		Set proxy options for certain HTTP client services.
Registration Authority Options		Set registration authority configuration options.
Reset Database	Database initialized: 2020-10-01 21:28:05	Rebuild the database and reset the system.
Restore	Previous restore: 2020-10-13 18:35:50	Restore database including keys and certificates.
SCSA Options		Set SCSA configuration options.
SNMP Options		Configure SNMP walk options.
System Key Options		Manage system keys.
Update Firmware		Update the encryption card firmware.
Upload Feature Request		Upload new application abilities
Vault API Options		Set options for interacting with an external Vault service.
Web Server Configuration		Set web server configuration options.

Group: Admin Group Users: Admin1,Admin2

Check the "Enable Vault Service" box.



Vault Options

☒ Enable Vault Service

Vault API URL: https://10.0.5.118:8210/v1

Namespace:

Certificate Mount Point: cert

Userpass Mount Point: userpass

Secret Mount Point: secret

Secret Path:

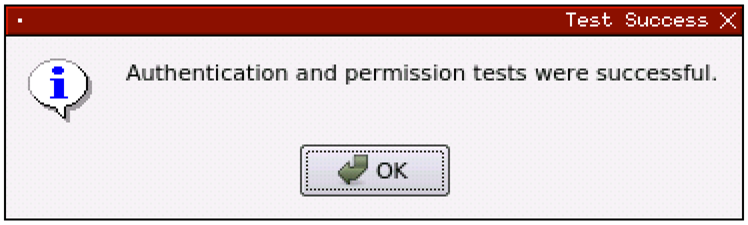
Require User Path: ☐

Timeout: 5

Credential: vault_certificate_authentication

Set the Vault API URL to "https://10.0.5.118:8210/v1" and select the Vault Certificate Authentication Cloud Credential created in the previous step.

The rest of the fields can be left as their default values. Click "Test Configuration...".



Test Success

Authentication and permission tests were successful.

If all of the configuration steps were completed properly the test should result in a success.

[6] OFFLOADING RANDOMLY GENERATED PKCS #12 PASSPHRASES TO VAULT

In this section, two examples will be covered in which Futurex APIs are invoked to request an X.509 certificate and associated key pair, which is issued as a PKCS #12 file. Then, the passphrase that was generated for decrypting the PKCS #12 file is automatically offloaded to Vault by the KMES Series 3.

The first example will involve running the RAUP Excrypt command while connected to the System/Host API port on the KMES via OpenSSL.

The second example will involve sending a POST request to the KMES RESTful API using the Postman application.

[6.1] REQUIRED SETUP ON THE KMES SERIES 3

Before attempting either of the examples, the following must be set up on the KMES Series 3:

1. Create a Signing Approval Group.
2. Create a CA tree (**IMPORTANT:** Ensure that PMK is chosen as the major key for all certificates created in this CA tree)
3. Add an Issuance Policy to the leaf certificate in the CA tree (**IMPORTANT:** In the *X.509* tab of the *Issuance Policy* dialog, ensure that all of the permission boxes are checked and that the Signing Approval Group created in step 1 is selected)
4. Create a new User Group with all of the "Manage certificates", "Manage keys", and "Perform cryptographic operations" permissions, and set the number of users required to log in to "1".
5. Add one user to the User Group that was created in the previous step.
6. Give the created User Group "Use" permissions on all of the following:
    - The Signing Approval Group created in step 1
    - The entire CA tree created in step 2
    - The Cloud Credential that is being used in the *Vault API Options* menu

NOTE: Please refer to the KMES Series 3 user guide for information about completing the actions above.

[6.2] EXAMPLE - EXCRYPT COMMAND RAUP

First, ensure that the RAUP command is enabled in the *Host API Options* menu.

Then, connect to the System/Host API port on the KMES via OpenSSL.

```
$ openssl s_client -connect 10.0.8.28:2001 -cert signed-client-cert.pem -key private-key.pem -
CApath . -CAfile chain.pem
```

NOTE: The System/Host API connection pair on the KMES Series 3 must be configured so that this OpenSSL connection will work. Instructions for setting this up are outside the scope of this course. Please refer to the KMES Series 3 user guide for more information.

Once connected successfully, run the RKLO Excrypt command twice to login in with the default Admin users.

```
[AORKLO;DAAdmin1;CHsafest;]
[AORKLO;ANY;UL1;UT2;RL1;]

[AORKLO;DAAdmin2;CHsafest;]
[AORKLO;ANY;UL2;UT2;RL0;]
```

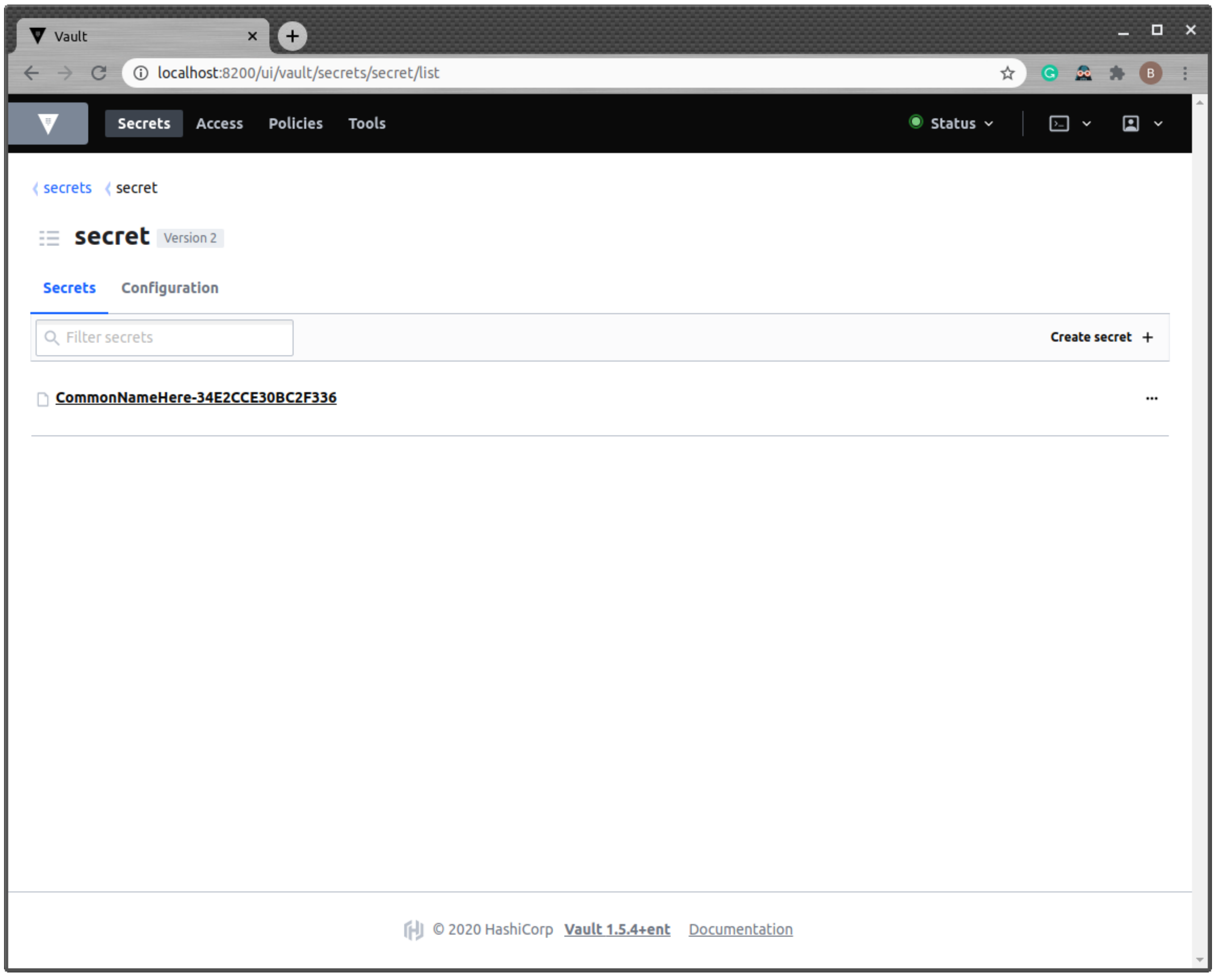
Now that we're logged in, we can run the RAUP command to upload a new X.509 PKI request.

```
[AORAUP;CADemo CA;RTDemo Sub Cert;NATestUpload;HASHA256;GNDemo Approval Group;ENExample TLS Cer-
tificate;SN{2.5.4.3,12,436F6D6D6F6E4E616D6548657265},{1.3.3.7,19,30313233};KTRSA 2048;MP1;]

[AORAUP;ANY;AP1;ID34E2CCE30BC2F336;PW567E463B516F3120246C265446366A585D63636A794342215F344B7D43266F-
765D454C3B5741307068614F657A487A2234692E51754B5C46342477282856253C;]
```

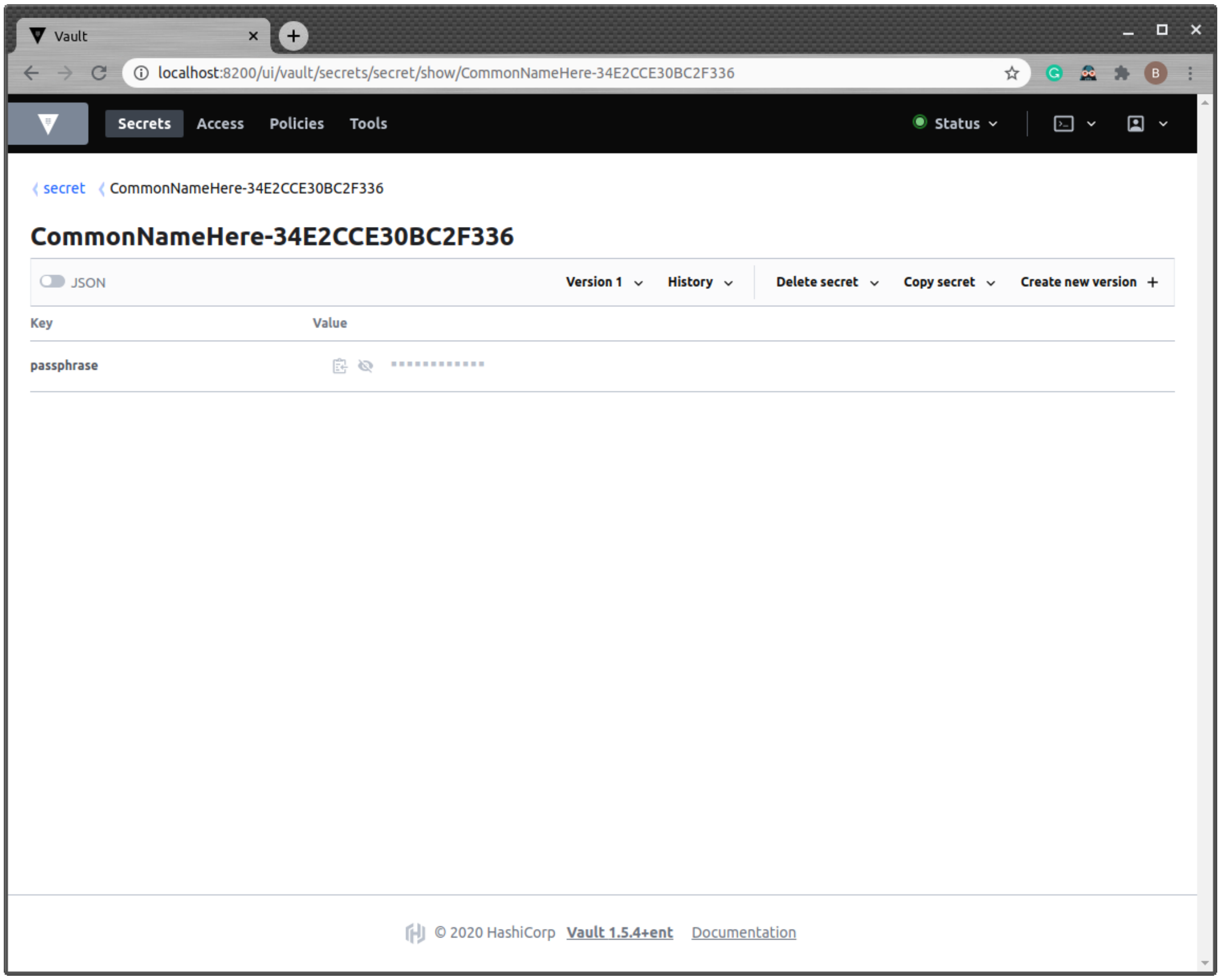
If the command succeeds, as it does above, a new X.509 PKI, issued as a PKCS #12 file, will be generated on the KMES Series 3, and the passphrase for the PKCS #12 file will be stored in Vault.

NOTE: The RAUP command above uses the CA tree that was created in section 5.1. For more information about the RAUP command tags, please refer to API documentation in the Futurex Portal.

Now, if we log in to the Vault UI and navigate into the "secret/" folder we'll see the passphrase secret was added successfully.



Click on the "CommonNameHere-34E2CCE30BC2F336" secret.



Key	Value
passphrase	•••••••••••••

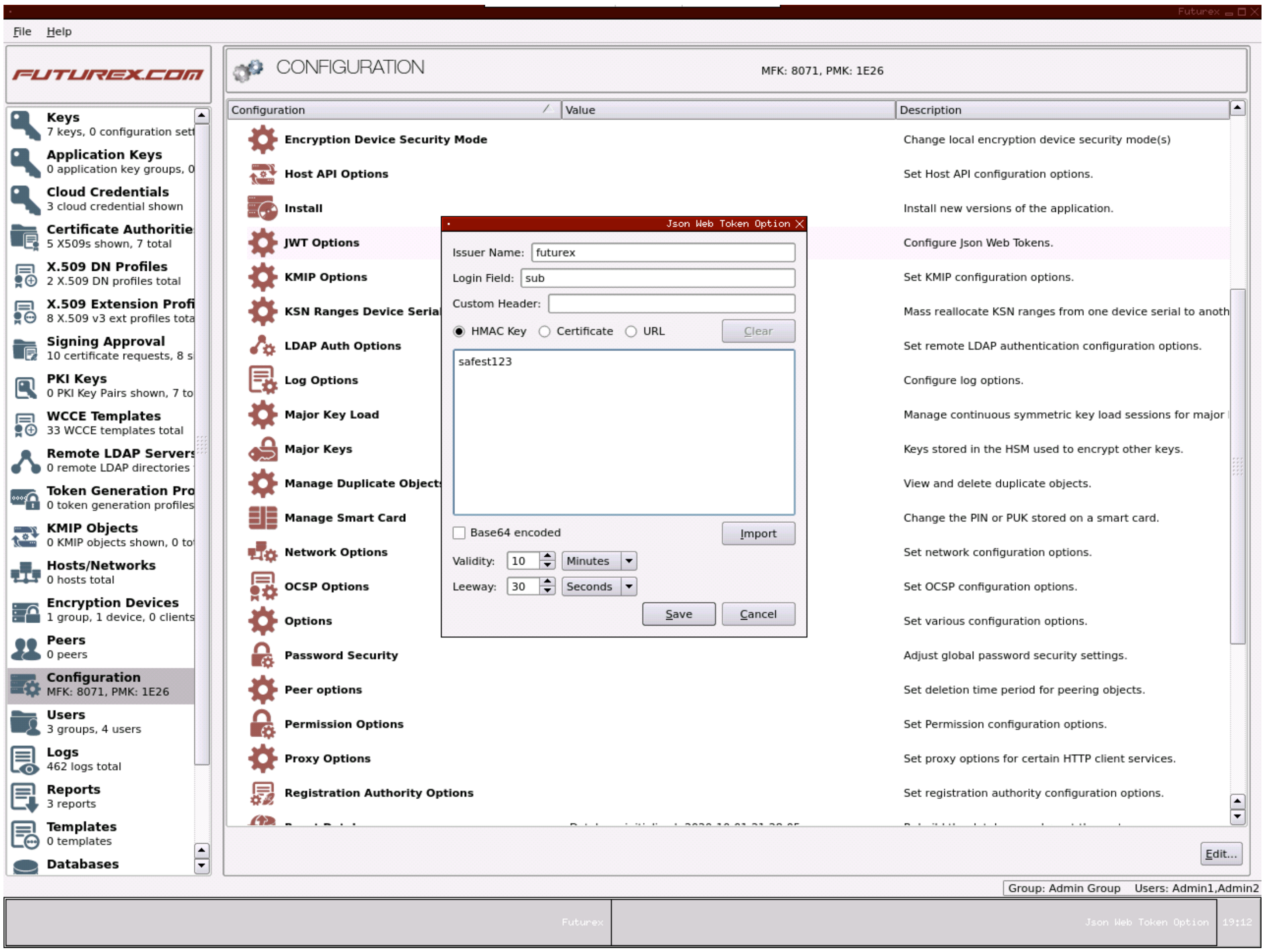
We see that the *Key* is "passphrase" and the *Value* is the passphrase for the PKCS #12 file that was generated on the KMES Series 3.

[6.3] EXAMPLE - POST REQUEST TO THE KMES RESTFUL API USING POSTMAN

This example requires an additional configuration to be made on the KMES Series 3.

Configure JWT Options

Navigate to *Configuration -> JWT Options*.



File Help

CONFIGURATION MFK: 8071, PMK: 1E26

Keys	7 keys, 0 configuration sett
Application Keys	0 application key groups, 0
Cloud Credentials	3 cloud credential shown
Certificate Authoritie	5 X509s shown, 7 total
X.509 DN Profiles	2 X.509 DN profiles total
X.509 Extension Profi	8 X.509 v3 ext profiles tota
Signing Approval	10 certificate requests, 8 s
PKI Keys	0 PKI Key Pairs shown, 7 to
WCCE Templates	33 WCCE templates total
Remote LDAP Servers	0 remote LDAP directories
Token Generation Pro	0 token generation profiles
KMIP Objects	0 KMIP objects shown, 0 to
Hosts/Networks	0 hosts total
Encryption Devices	1 group, 1 device, 0 clients
Peers	0 peers
Configuration	MFK: 8071, PMK: 1E26
Users	3 groups, 4 users
Logs	462 logs total
Reports	3 reports
Templates	0 templates
Databases	

Configuration	Value	Description
Encryption Device Security Mode		Change local encryption device security mode(s)
Host API Options		Set Host API configuration options.
Install		Install new versions of the application.
JWT Options		Configure Json Web Tokens.
KMIP Options		Set KMIP configuration options.
KSN Ranges Device Serial		Mass reallocate KSN ranges from one device serial to anoth
LDAP Auth Options		Set remote LDAP authentication configuration options.
Log Options		Configure log options.
Major Key Load		Manage continuous symmetric key load sessions for major
Major Keys		Keys stored in the HSM used to encrypt other keys.
Manage Duplicate Objects		View and delete duplicate objects.
Manage Smart Card		Change the PIN or PUK stored on a smart card.
Network Options		Set network configuration options.
OCSP Options		Set OCSP configuration options.
Options		Set various configuration options.
Password Security		Adjust global password security settings.
Peer options		Set deletion time period for peering objects.
Permission Options		Set Permission configuration options.
Proxy Options		Set proxy options for certain HTTP client services.
Registration Authority Options		Set registration authority configuration options.

Json Web Token Option

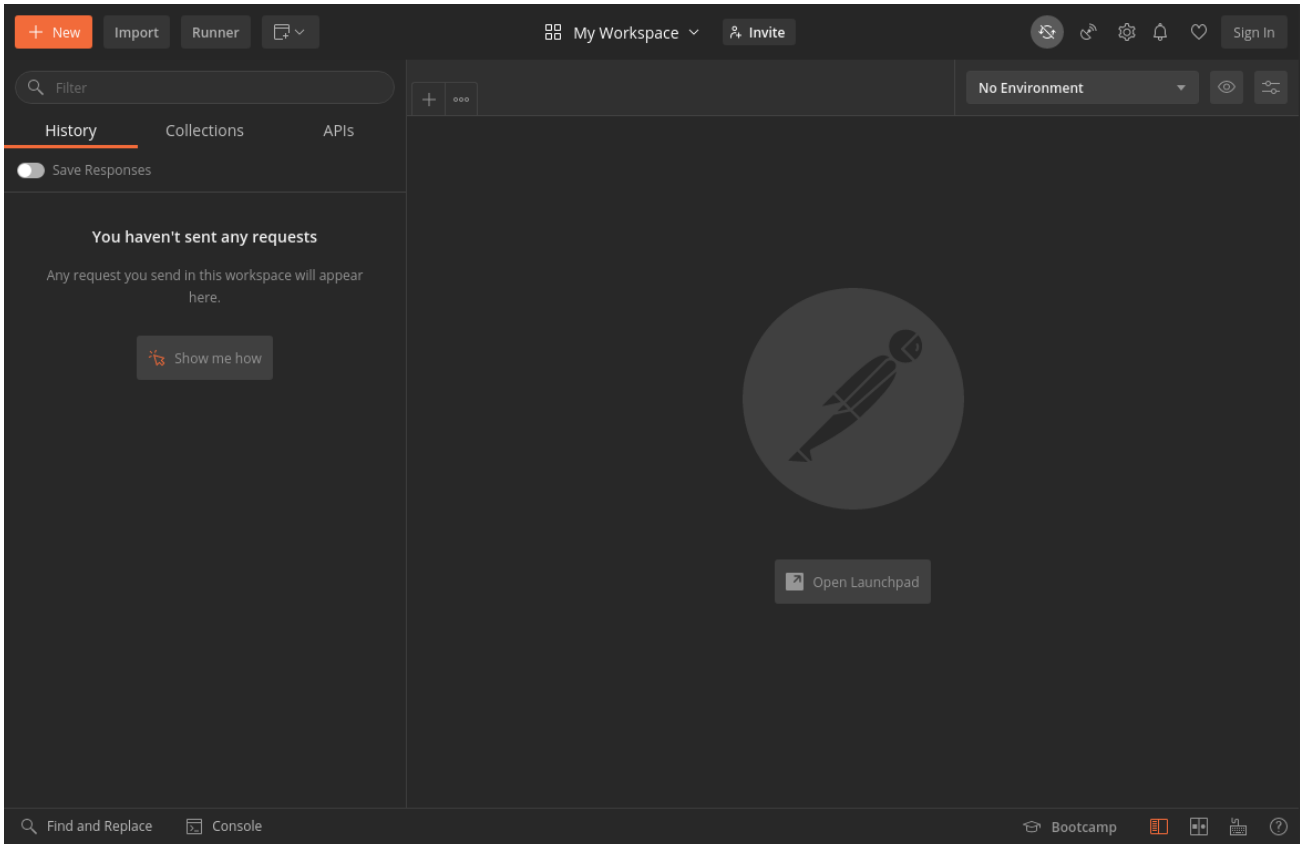
Issuer Name: futurex
Login Field: sub
Custom Header:
[✓ HMAC Key] Certificate URL
safest123
Base64 encoded
Validity: 10 Minutes
Leeway: 30 Seconds

In the *Json Web Token Option* dialog, set the *Issuer Name* to "futurex" and set "safest123" as an HMAC Key password. The remaining fields can be left as their default values. Then, click "Save".

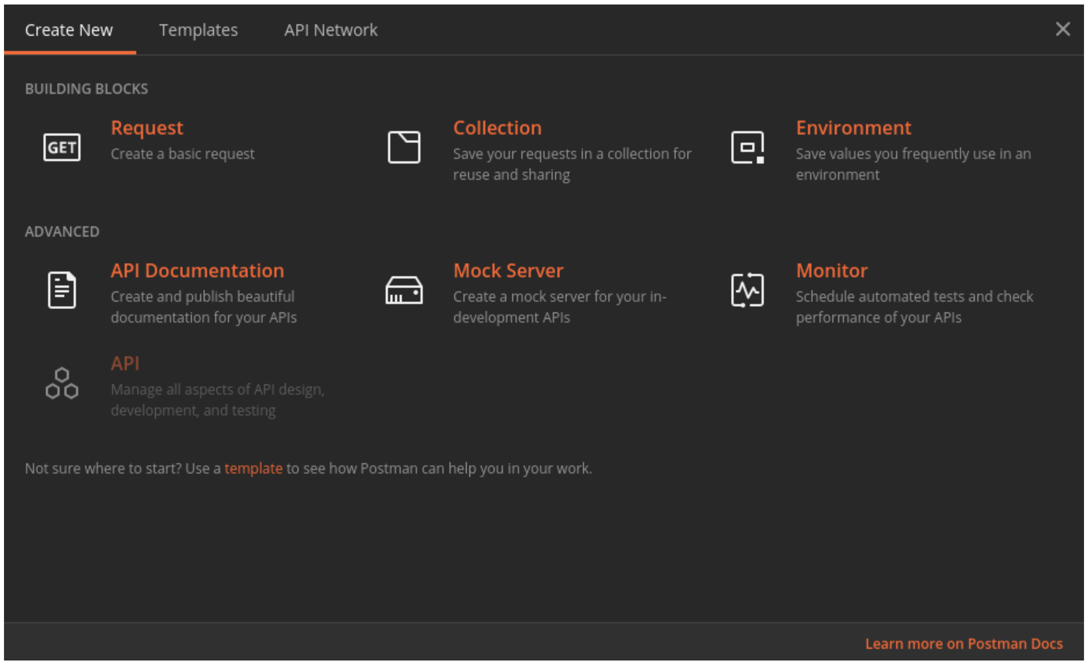
Sending a POST request to the KMES RESTful API using Postman

NOTE: To perform the steps that follow, you must have the Postman application installed on your local computer. The same concepts would apply if you wanted to use cURL rather than Postman. However, to use cURL, you would need to generate the JWT token elsewhere (i.e., a website such as https://jwt.io/), whereas, with Postman, we'll be able to generate a JWT token "on-the-fly" when the POST request is sent.

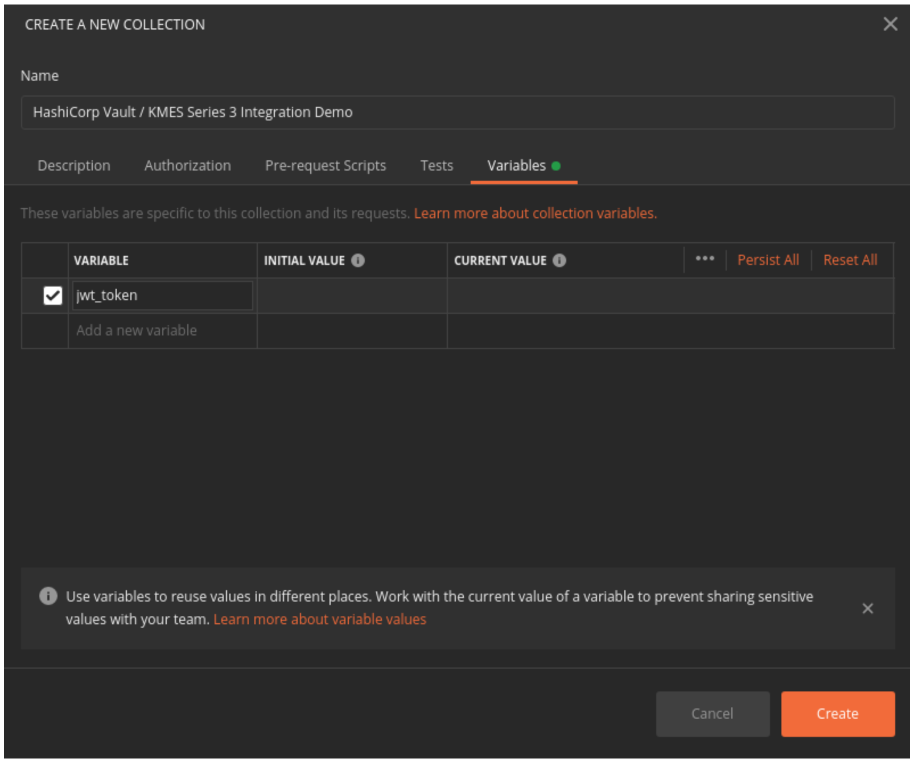
Start the Postman application, then click the orange "New" button in the top left area of the page.



Select "Collection".


Create New
Templates
API Network
BUILDING BLOCKS
Request
Create a basic request
Collection
Save your requests in a collection for reuse and sharing
Environment
Save values you frequently use in an environment
ADVANCED
API Documentation
Create and publish beautiful documentation for your APIs
Mock Server
Create a mock server for your in-development APIs
Monitor
Schedule automated tests and check performance of your APIs
API
Manage all aspects of API design, development, and testing
Not sure where to start? Use a template to see how Postman can help you in your work.
Learn more on Postman Docs

Set any name for the Collection, then navigate to the *Variables* tab and set "jwt_token" in the VARIABLE field, then click "Create" (nothing else needs to be set in this dialog).



CREATE A NEW COLLECTION

Name: HashiCorp Vault / KMES Series 3 Integration Demo

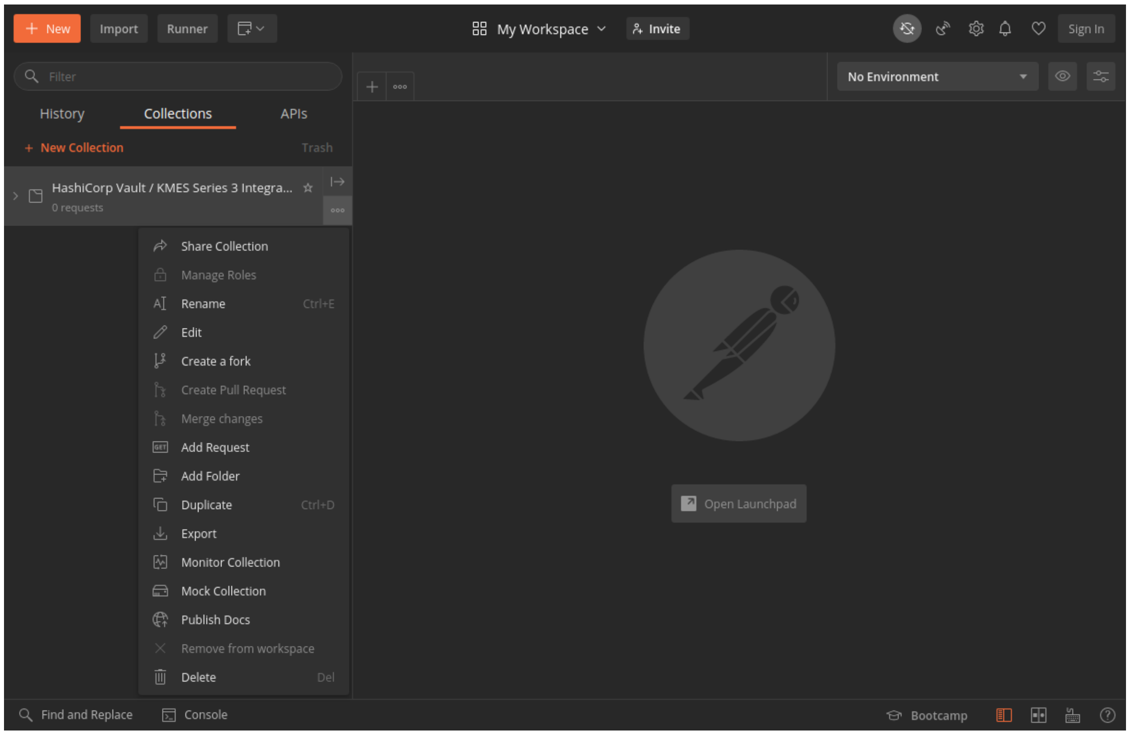
Description | Authorization | Pre-request Scripts | Tests | Variables

These variables are specific to this collection and its requests. Learn more about collection variables.

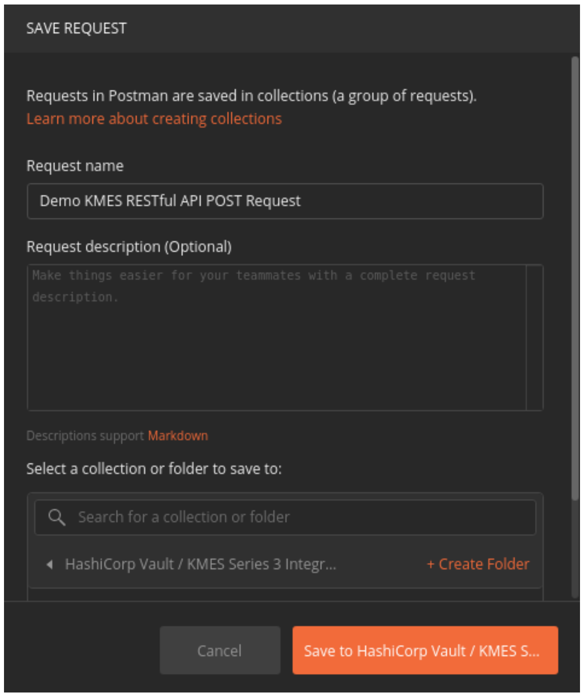
	VARIABLE	INITIAL VALUE	CURRENT VALUE
✓	jwt_token		
	Add a new variable		

Use variables to reuse values in different places. Work with the current value of a variable to prevent sharing sensitive values with your team. Learn more about variable values

Right-click on the 3 dots in the new Collection and select "Add Request".



Set any name for the request. The most important part is that the request is associated with the newly created Collection. Save the request.



SAVE REQUEST

Requests in Postman are saved in collections (a group of requests).
Learn more about creating collections

Request name: Demo KMES RESTful API POST Request

Request description (Optional): Make things easier for your teammates with a complete request description.

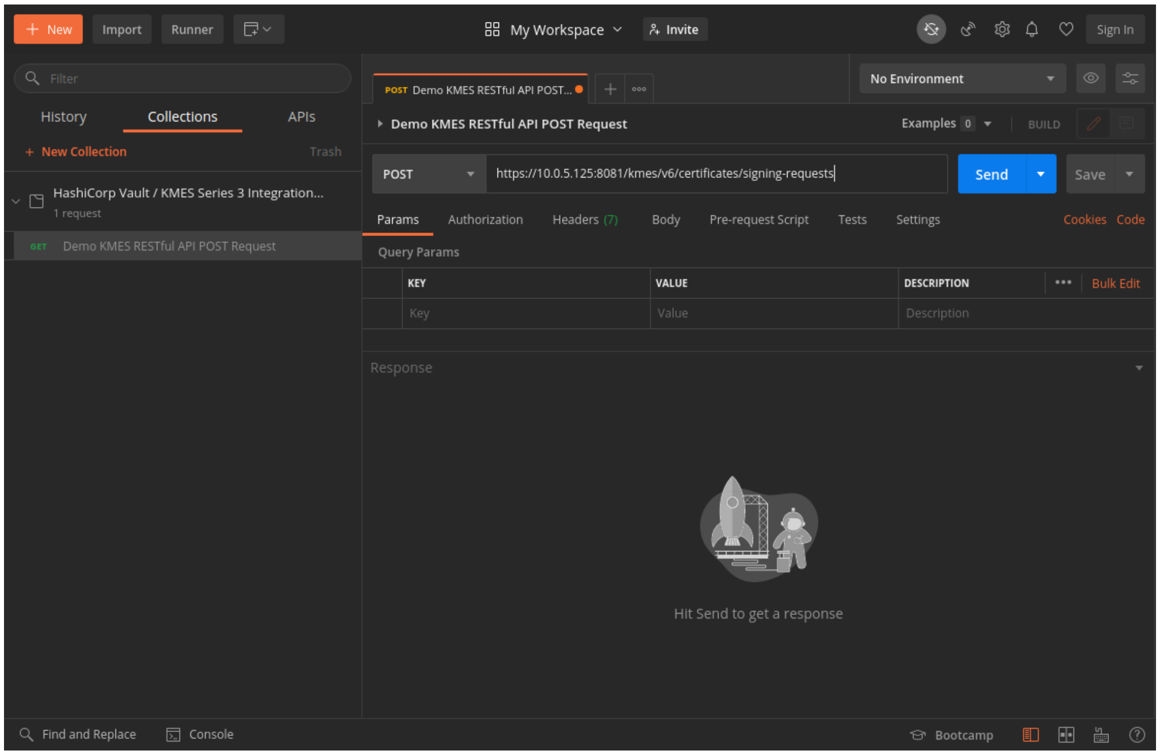
Descriptions support Markdown

Select a collection or folder to save to:

Search for a collection or folder

HashiCorp Vault / KMES Series 3 Integr... + Create Folder

Expand the Collection folder and select the request that was just created. Set the request URL to "https://10.0.5.125:8081/kmes/v6/certificates/signing-requests" and change the request type to "POST" in the drop-down.



HashiCorp Vault / KMES Series 3 Integration...
1 request
GET Demo KMES RESTful API POST Request

POST Demo KMES RESTful API POST...

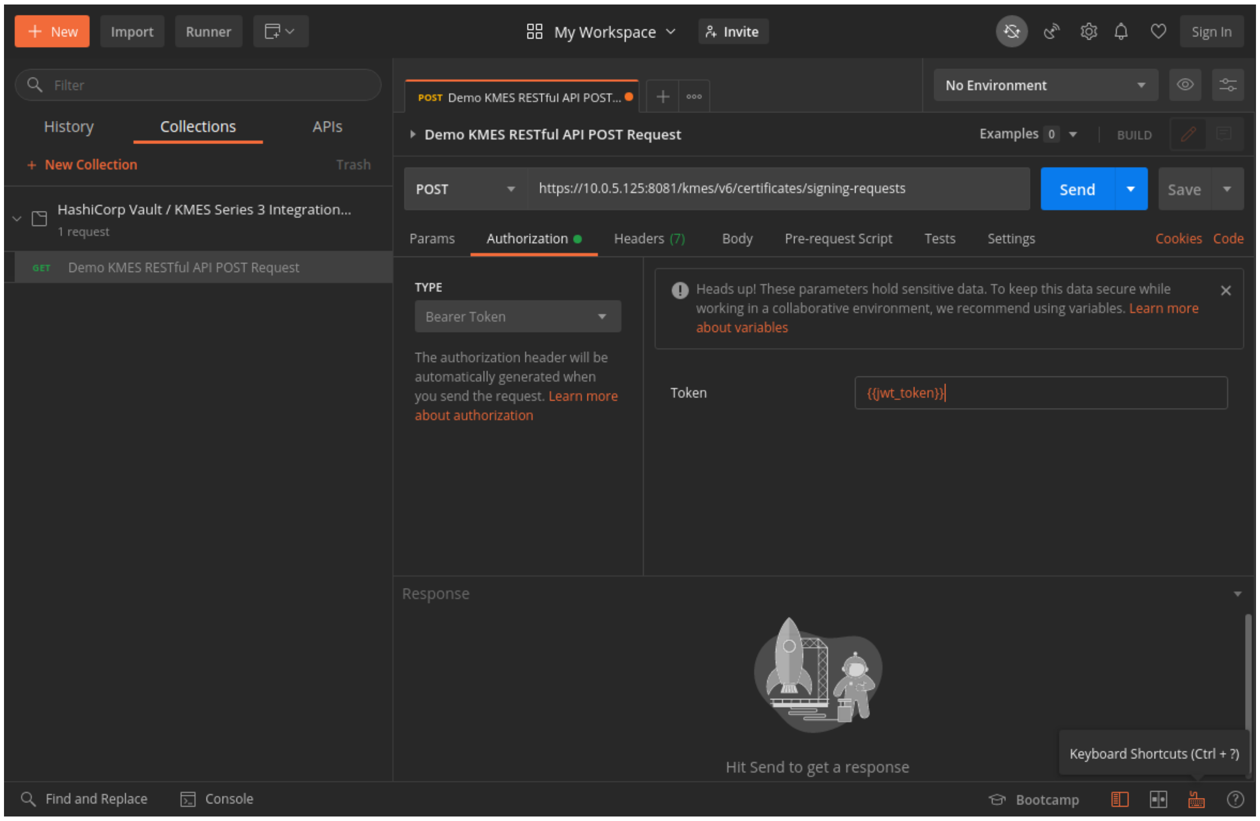
No Environment

Demo KMES RESTful API POST Request

POST https://10.0.5.125:8081/kmes/v6/certificates/signing-requests

KEY	VALUE	DESCRIPTION
Key	Value	Description

Switch to the *Authorization* tab, set the TYPE to "Bearer Token", then set the Token value to "{{jwt_token}}".



Demo KMES RESTful API POST Request

POST https://10.0.5.125:8081/kmes/v6/certificates/signing-requests

TYPE: Bearer Token

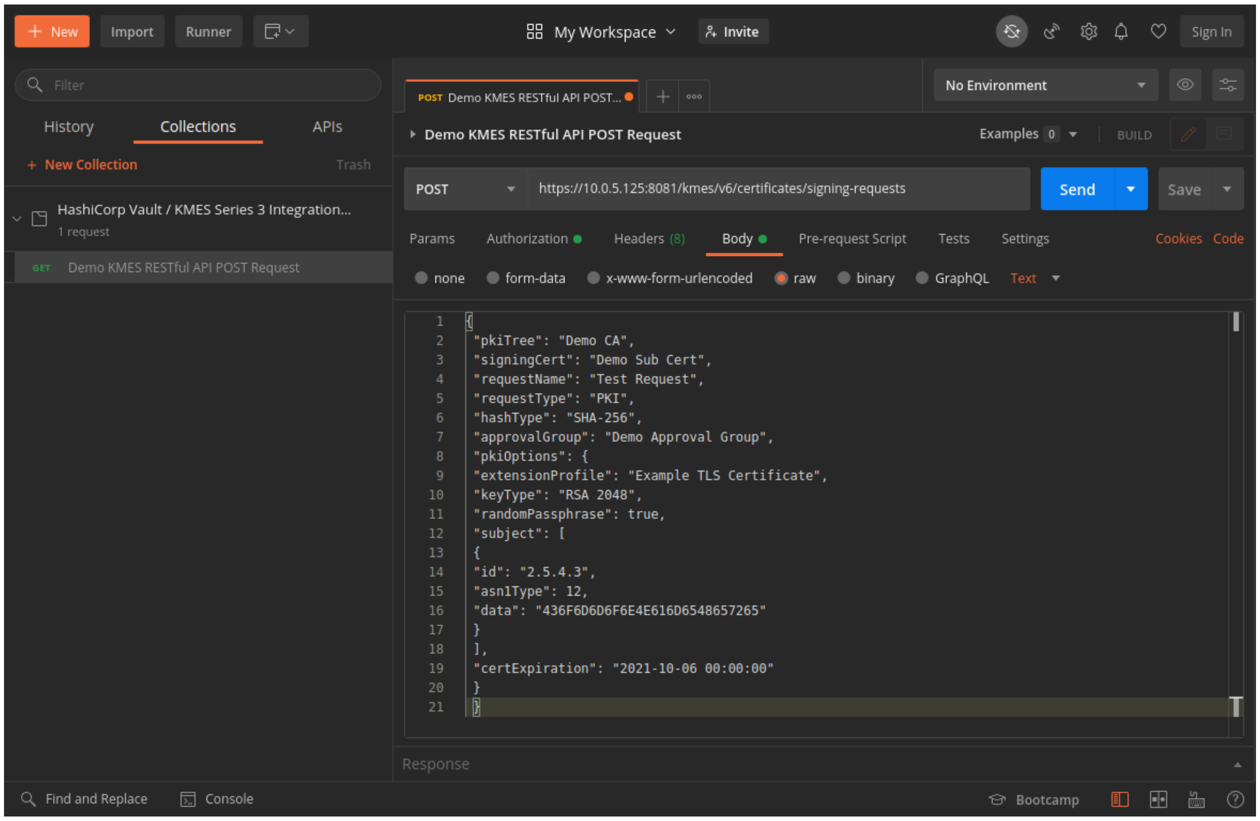
The authorization header will be automatically generated when you send the request. Learn more about authorization

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables

Token: {{jwt_token}}

Hit Send to get a response

Navigate to the *Body* tab, select the "raw" bullet, then paste in the JSON shown below.



Demo KMES RESTful API POST Request

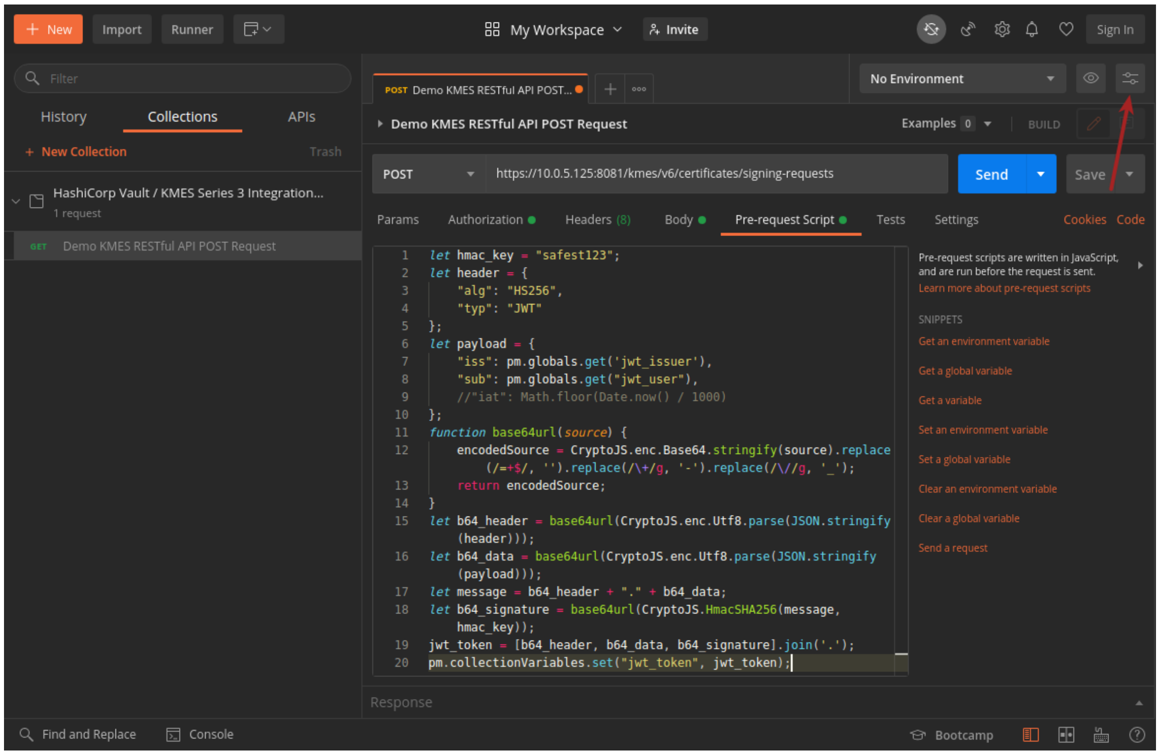
POST https://10.0.5.125:8081/kmes/v6/certificates/signing-requests

none | form-data | x-www-form-urlencoded | [✓ raw] | binary | GraphQL | Text

```
{
"pkiTree": "Demo CA",
"signingCert": "Demo Sub Cert",
"requestName": "Test Request",
"requestType": "PKI",
"hashType": "SHA-256",
"approvalGroup": "Demo Approval Group",
"pkiOptions": {
"extensionProfile": "Example TLS Certificate",
"keyType": "RSA 2048",
"randomPassphrase": true,
"subject": [
{
"id": "2.5.4.3",
"asn1Type": 12,
"data": "436F6D6D6F6E4E616D6548657265"
}
],
"certExpiration": "2021-10-06 00:00:00"
}
}
```

The request's body is where we specify that we want the KMES Series 3 to create a new X.509 PKI. The CA tree that we built on the KMES is defined, along with several other parameters. One of the most important parameters to notice is the "randomPassphrase" value set to "true". The "randomPassphrase" parameter ensures that the PKI request is issued as PKCS #12. Then, the passphrase for the PKCS #12 file will be stored in Vault.

Navigate to the *Pre-request Script* tab and paste in the Javascript below. This code is responsible for generating the JWT token used for authentication to the KMES, on-the-fly. You'll notice that the value for "hmac_key" in line 1 matches what was set in the JWT Options menu on the KMES.



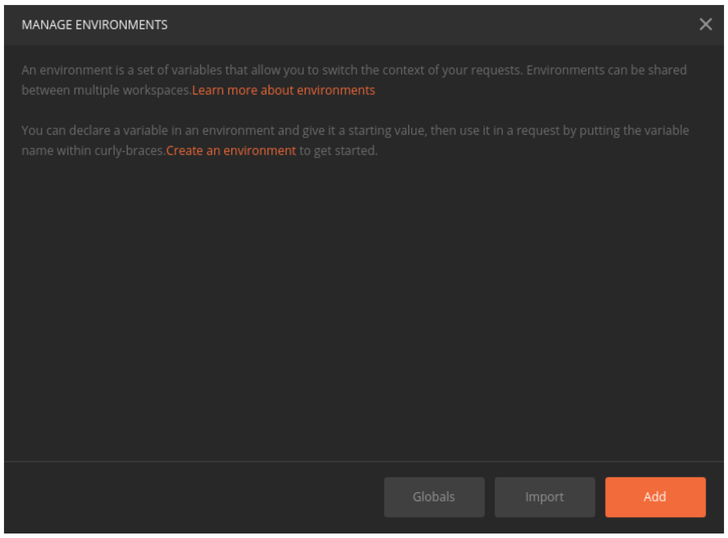
POST https://10.0.5.125:8081/kmes/v6/certificates/signing-requests

```
1   let hmac_key = "safest123";
2   let header = {
3       "alg": "HS256",
4       "typ": "JWT"
5   };
6   let payload = {
7       "iss": pm.globals.get('jwt_issuer'),
8       "sub": pm.globals.get("jwt_user"),
9       //"iat": Math.floor(Date.now() / 1000)
10  };
11  function base64url(source) {
12      encodedSource = CryptoJS.enc.Base64.stringify(source).replace
            (/=+$/, '').replace(/\+/g, '-').replace(/\//g, '_');
13      return encodedSource;
14  }
15  let b64_header = base64url(CryptoJS.enc.Utf8.parse(JSON.stringify
        (header)));
16  let b64_data = base64url(CryptoJS.enc.Utf8.parse(JSON.stringify
        (payload)));
17  let message = b64_header + "." + b64_data;
18  let b64_signature = base64url(CryptoJS.HmacSHA256(message,
        hmac_key));
19  jwt_token = [b64_header, b64_data, b64_signature].join('.');
20  pm.collectionVariables.set("jwt_token", jwt_token);
```

Pre-request scripts are written in JavaScript, and are run before the request is sent.
Learn more about pre-request scripts

SNIPPETS
Get an environment variable
Get a global variable
Get a variable
Set an environment variable
Set a global variable
Clear an environment variable
Clear a global variable
Send a request

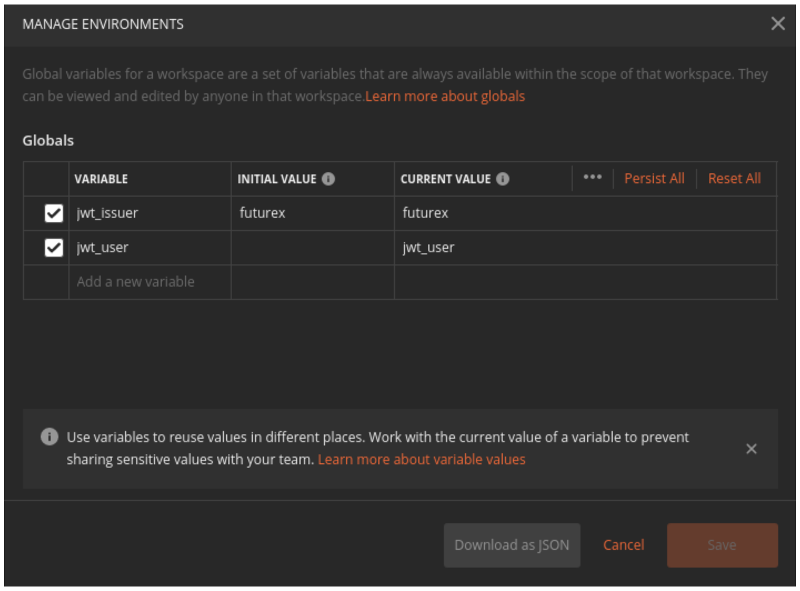
After pasting in the Javascript, click on the settings icon in the upper-right area of the page. Click the *Globals* button.


MANAGE ENVIRONMENTS

An environment is a set of variables that allow you to switch the context of your requests. Environments can be shared between multiple workspaces.Learn more about environments

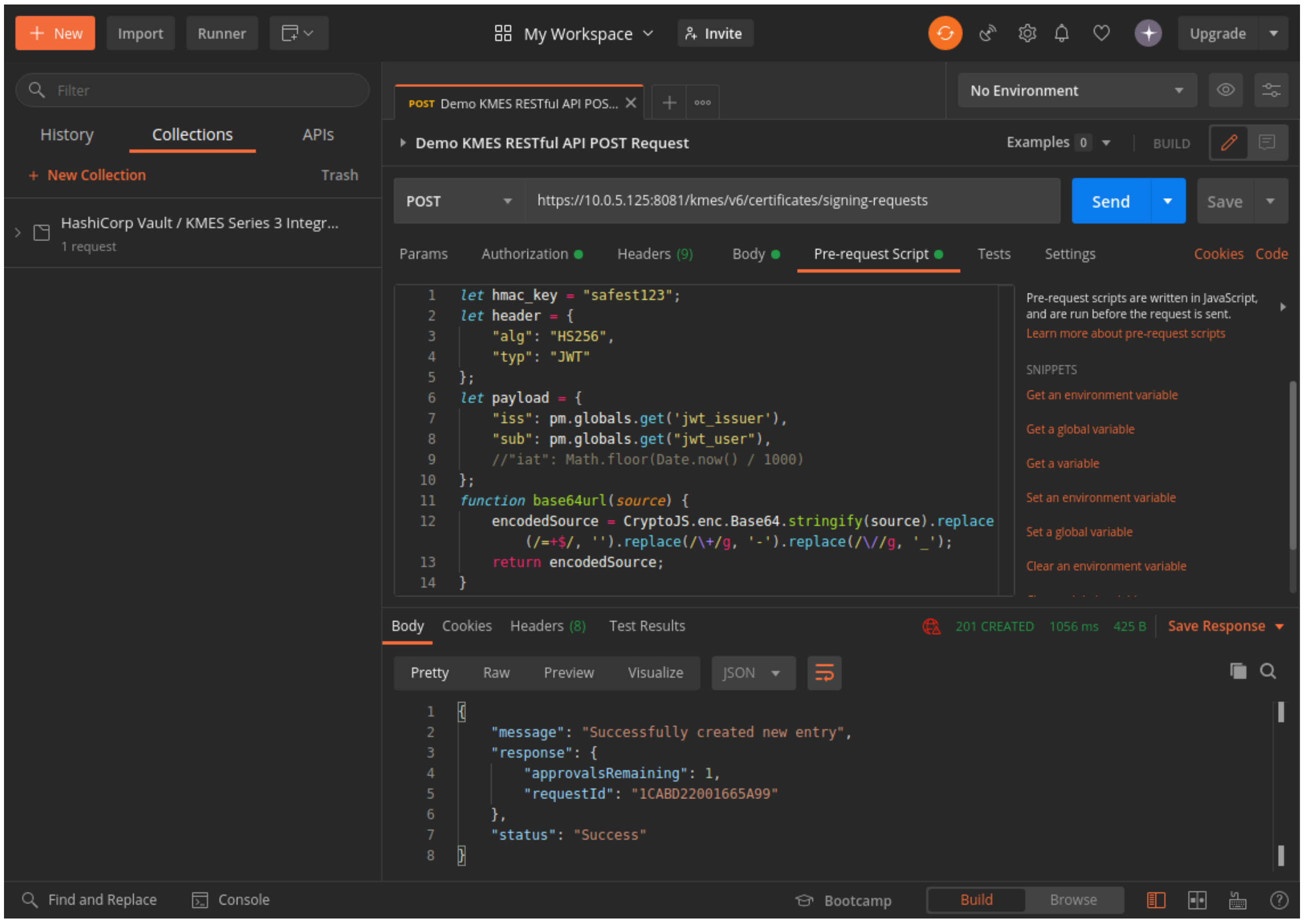
You can declare a variable in an environment and give it a starting value, then use it in a request by putting the variable name within curly-braces.Create an environment to get started.

Set two new variables exactly as they are shown below, then click "Save".


MANAGE ENVIRONMENTS
Global variables for a workspace are a set of variables that are always available within the scope of that workspace. They can be viewed and edited by anyone in that workspace. Learn more about globals
Globals
Persist All
Reset All
Use variables to reuse values in different places. Work with the current value of a variable to prevent sharing sensitive values with your team. Learn more about variable values
Download as JSON
Cancel
Save

	VARIABLE	INITIAL VALUE	CURRENT VALUE
✓	jwt_issuer	futurex	futurex
✓	jwt_user		jwt_user
	Add a new variable		

The setup is complete. Now click "Save" and then send the POST request.



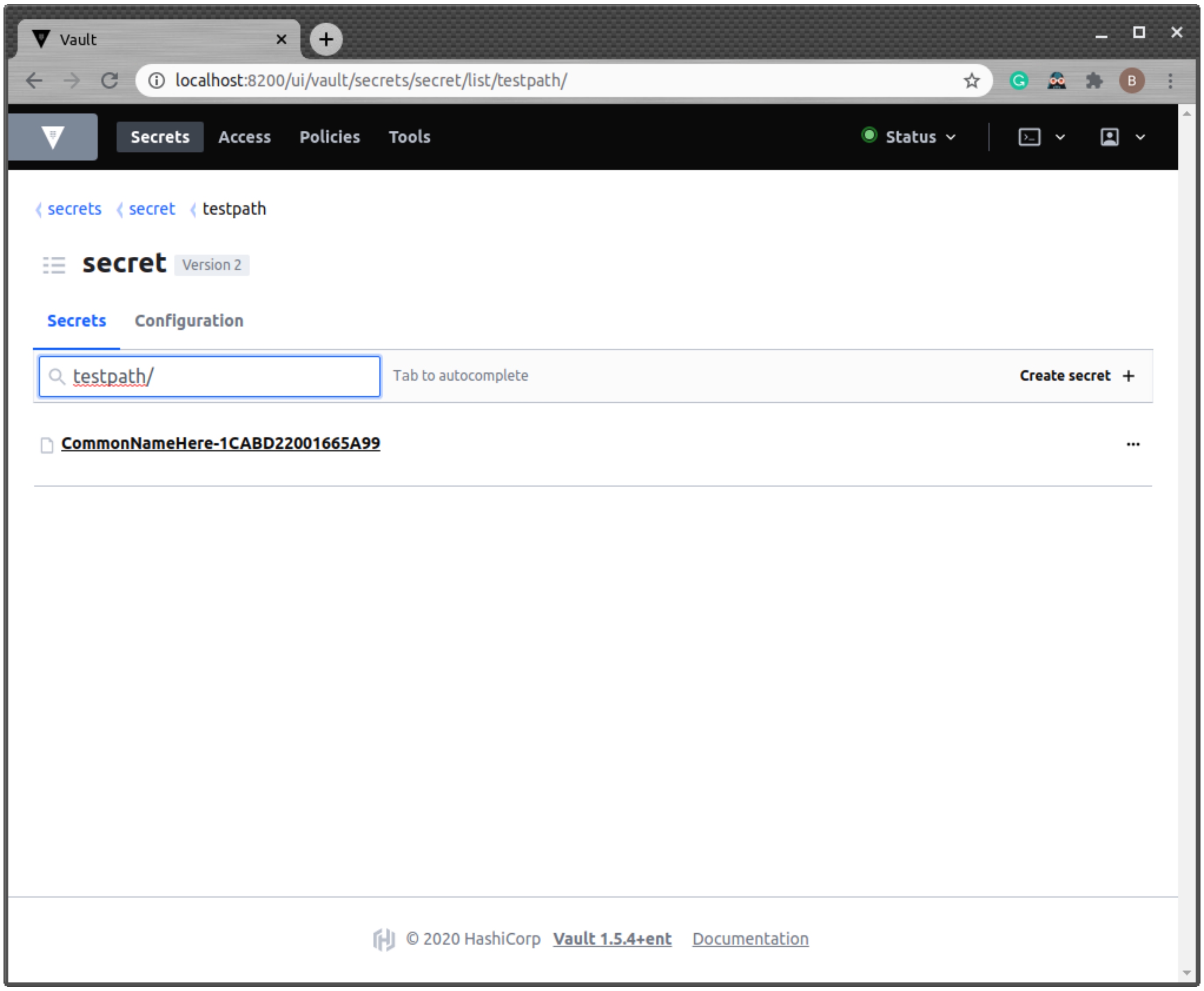
POST https://10.0.5.125:8081/kmes/v6/certificates/signing-requests

```
let hmac_key = "safest123";
let header = {
    "alg": "HS256",
    "typ": "JWT"
};
let payload = {
    "iss": pm.globals.get('jwt_issuer'),
    "sub": pm.globals.get("jwt_user"),
    //"iat": Math.floor(Date.now() / 1000)
};
function base64url(source) {
    encodedSource = CryptoJS.enc.Base64.stringify(source).replace
        (/=+$/, '').replace(/\+/g, '-').replace(/\//g, '_');
    return encodedSource;
}
```

```
{
    "message": "Successfully created new entry",
    "response": {
        "approvalsRemaining": 1,
        "requestId": "1CABD22001665A99"
    },
    "status": "Success"
}
```

If the request is successful, the body of the response will contain the message "Successfully created new entry", as it does above.

Now if you log back in to the Vault UI and go to "secret/testpath/" you will see a new entry, with the ID values after "CommonNameHere" matching the "requestId" value in the body of the response in Postman.



This confirms that the PKCS #12 passphrase was successfully sent from the KMES Series 3 to Vault for storage.

APPENDIX A: XCEPTIONAL SUPPORT

In today's high-paced environment, we know you are looking for timely and effective resolutions for your mission-critical needs. That is why our Xceptional Support Team does whatever it takes to ensure you have the best experience and support possible. Every time. Guaranteed.

- 24x7x365 mission critical support
- Level 1 to level 3 support
- Extremely knowledgeable subject matter experts

At Futurex, we strive to supply you with the latest data encryption innovations as well as our best-in-class support services. Our Xceptional Support Team goes above and beyond to meet your needs and provide you with exclusive services that you cannot find anywhere else in the industry.

- Technical Services
- Onsite Training
- Virtual Training
- Customized Consulting
- Customized Software Solutions
- Secure Key Generation, Printing, and Mailing
- Remote Key Injection
- Certificate Authority Services

Toll-Free: 1-800-251-5112

E-mail: support@futurex.com

ENGINEERING CAMPUS

864 Old Boerne Road

Bulverde, Texas, USA 78163

Phone: +1 830-980-9782

+1 830-438-8782

E-mail: info@futurex.com

XCEPTIONAL SUPPORT

24x7x365

Toll-Free: 1-800-251-5112

E-mail: support@futurex.com

SOLUTIONS ARCHITECT

E-mail: solutions@futurex.com