



# HASHICORP VAULT / KMES SERIES 3 PKCS #12 PASSWORD EXPORT

Integration Guide

**Applicable Devices:**

*KMES Series 3*



THIS DOCUMENT CONTAINS CONFIDENTIAL INFORMATION PROPRIETARY TO FUTUREX, LP. ANY UNAUTHORIZED USE, DISCLOSURE, OR DUPLICATION OF THIS DOCUMENT OR ANY OF ITS CONTENTS IS EXPRESSLY PROHIBITED.

## TABLE OF CONTENTS

[1] OVERVIEW OF THE HASHICORP VAULT / KMES SERIES 3 SECRET EXPORT INTEGRATION .....	3
[1.1] ABOUT HASHICORP VAULT .....	3
[1.2] PURPOSE OF THE INTEGRATION .....	3
[2] PREREQUISITES .....	4
[3] VAULT SETUP AND CONFIGURATION .....	5
[3.1] DOWNLOAD AND INSTALL VAULT .....	5
[3.2] CONFIGURE VAULT .....	5
[3.3] START THE DEV SERVER .....	7
[3.4] ACCESSING THE VAULT UI .....	9
[3.5] MODIFY THE DEFAULT ACL POLICY .....	10
[4] SETTING UP AUTHENTICATION BETWEEN THE KMES SERIES 3 AND VAULT .....	11
[4.1] USERPASS AUTH METHOD .....	11
[4.2] TLS CERTIFICATES AUTH METHOD .....	17
[5] OFFLOADING RANDOMLY GENERATED PKCS #12 PASSPHRASES TO VAULT .....	27
[5.1] REQUIRED SETUP ON THE KMES SERIES 3 .....	27
[5.2] EXAMPLE - EXCRYPT COMMAND RAUP .....	27
[5.3] EXAMPLE - POST REQUEST TO THE KMES RESTFUL API USING POSTMAN .....	30
APPENDIX A: XCEPTIONAL SUPPORT .....	38

## [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] 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

## [3] VAULT SETUP AND CONFIGURATION

### [3.1] 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
```

### [3.2] 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>

### [3.3] 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.

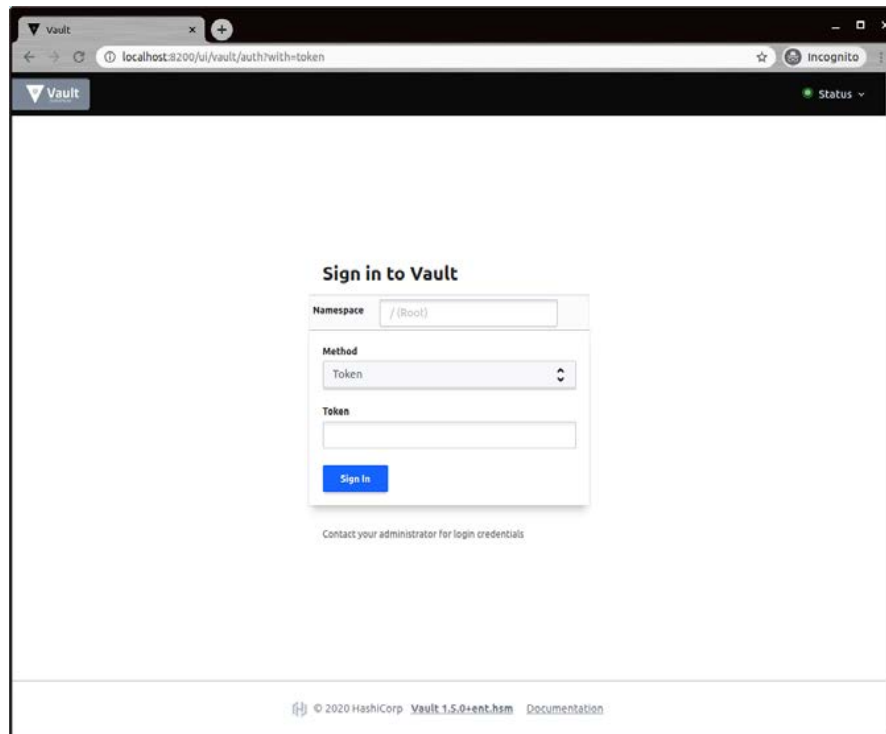
For more information about how the dev server works, please refer to <https://www.vaultproject.io/docs/concepts/dev-server>.

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>).

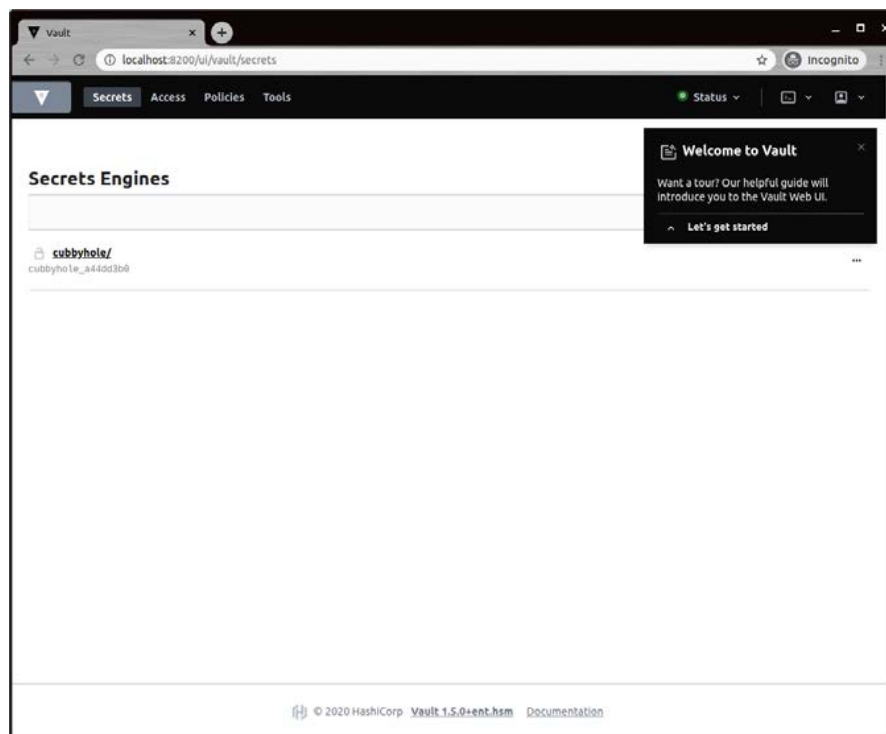


### [3.4] ACCESSING THE VAULT UI

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

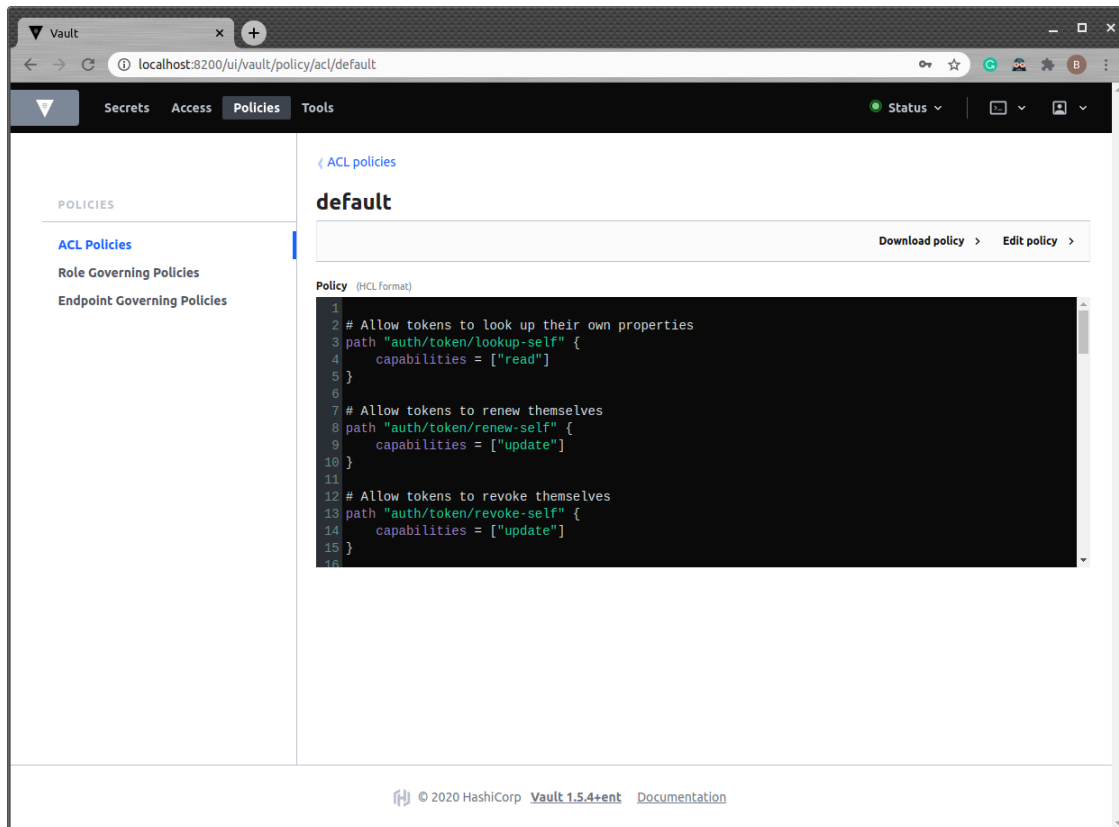


Copy and paste in the Root Token that was output from the **vault server** command into the "Token" field, then click "Sign In".



### [3.5] MODIFY THE DEFAULT ACL POLICY

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



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.

## [4] 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.

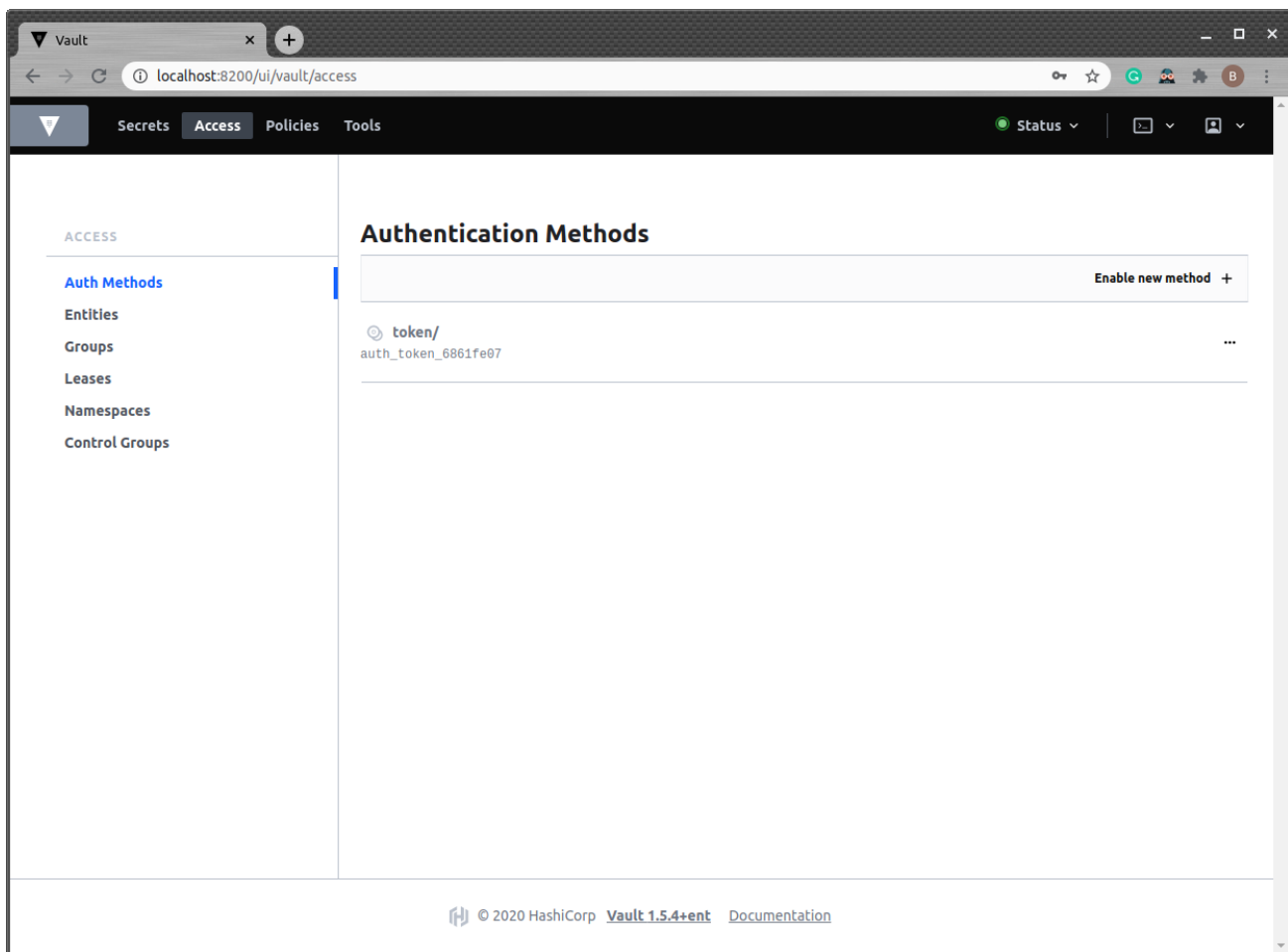
### [4.1] USERPASS AUTH METHOD

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

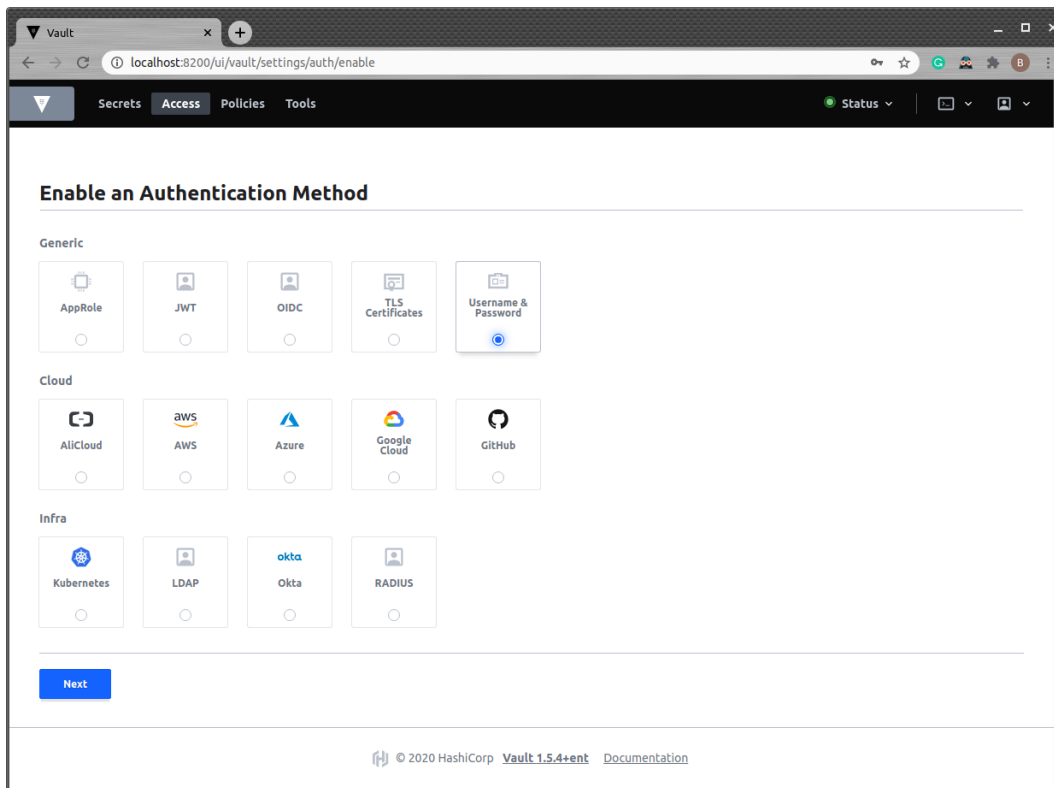
#### Configuring userpass authentication in Vault

##### Method 1 - Using the Vault UI

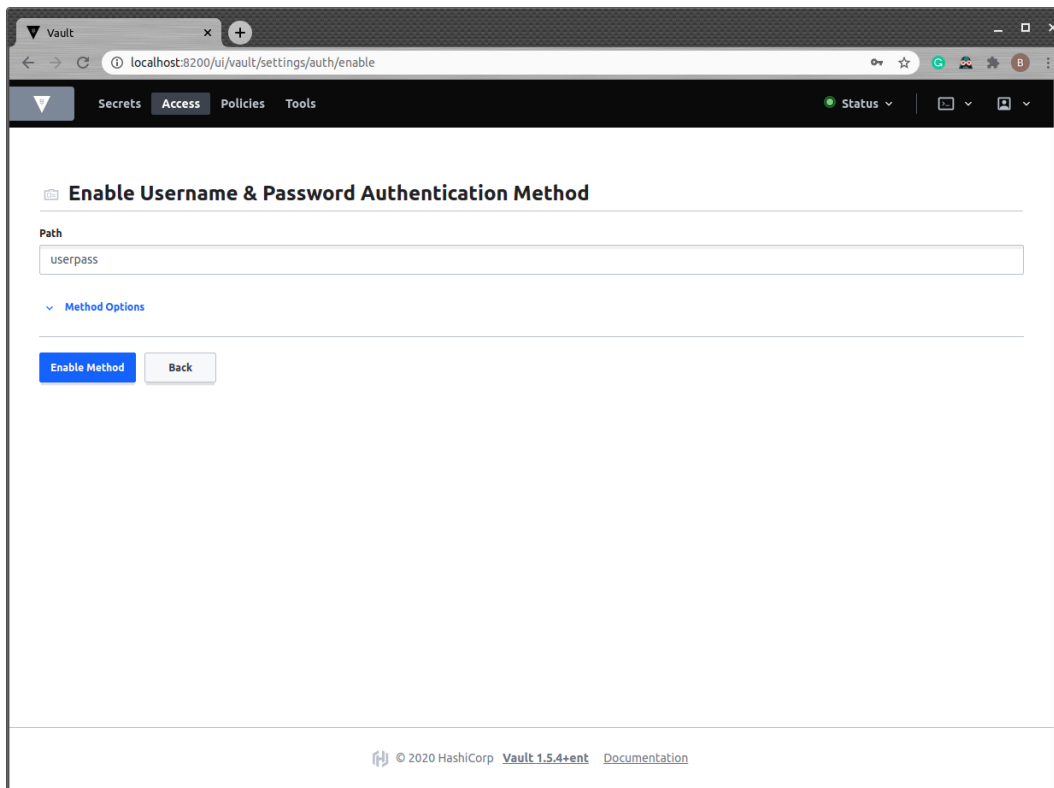
Navigate to the Access page in the Vault UI, then select "Enable new method".



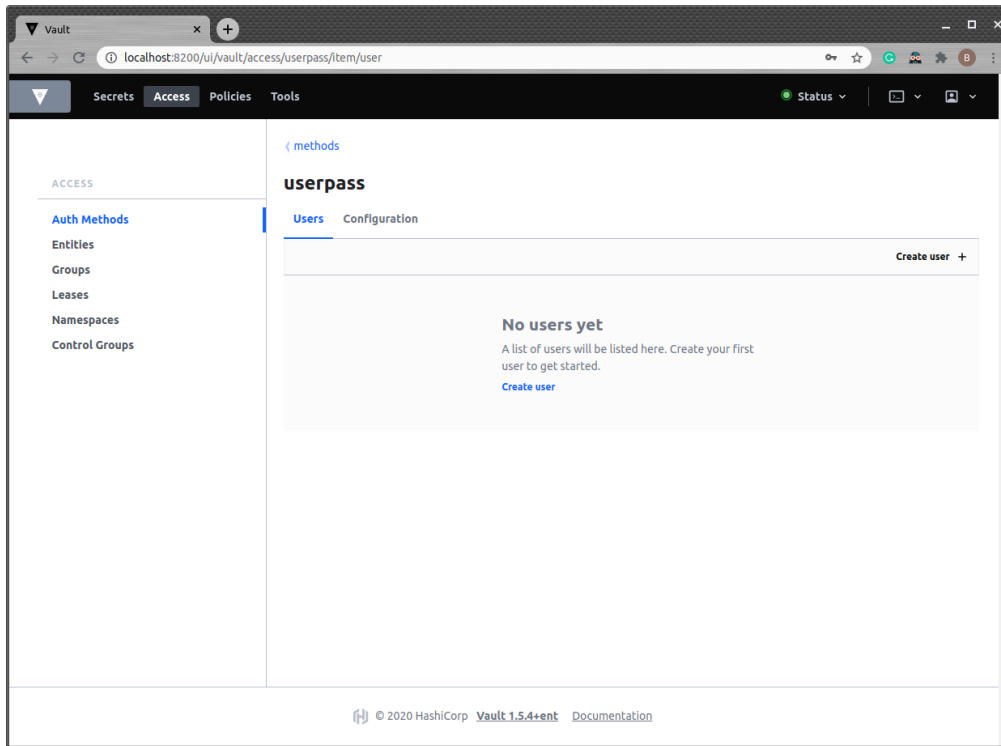
Select the "Username & Password" authentication method, then click "Next".



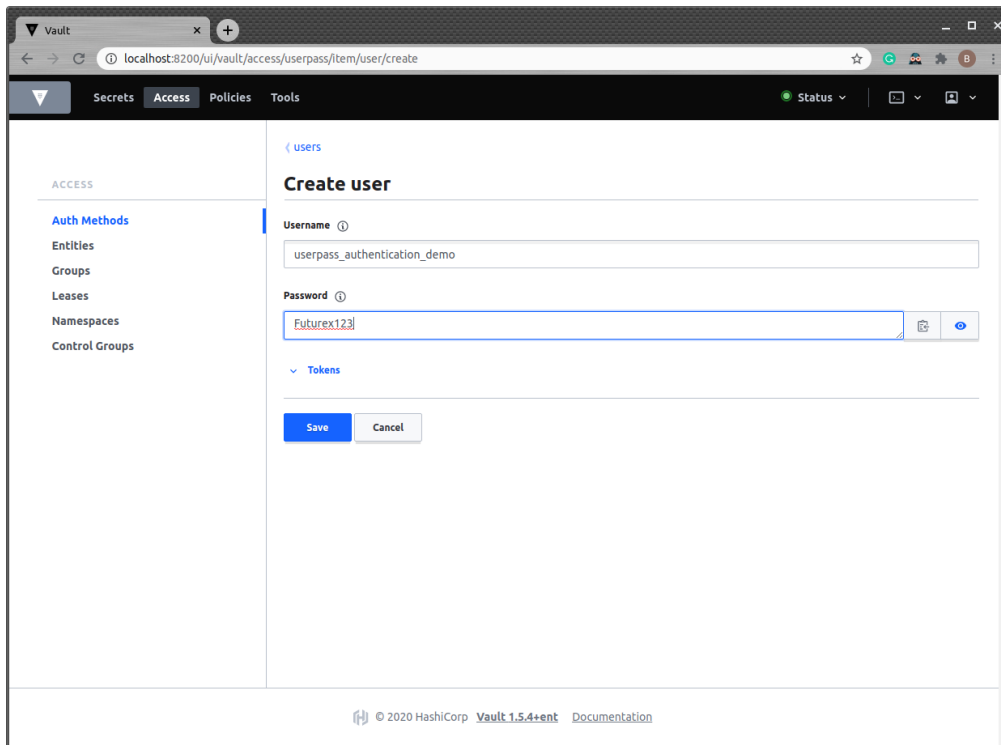
Leave the path as the default value, "userpass", then click "Enable Method".



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



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



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
```

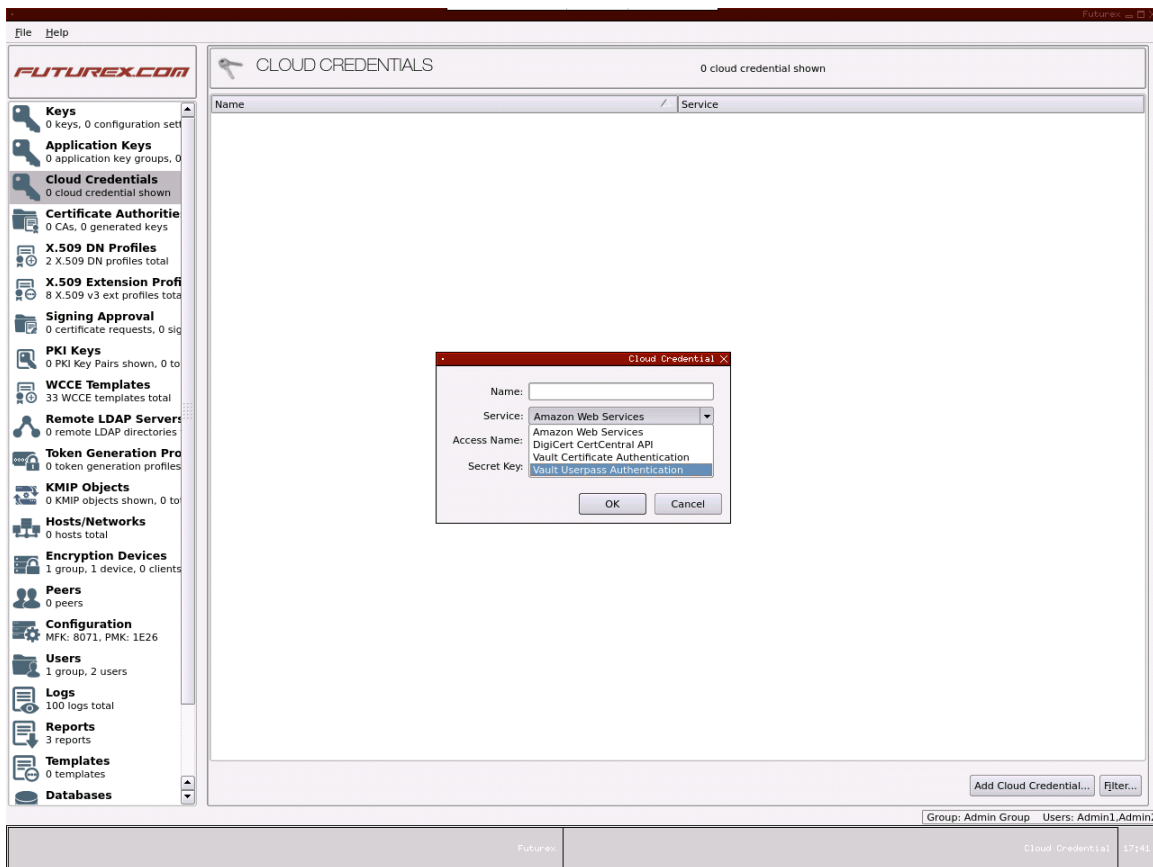
This creates a new user "userpass\_authentication\_demo" with the password "Futurex123" that will be associated with the "admins" policy. This is the only configuration necessary.

### 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".



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.

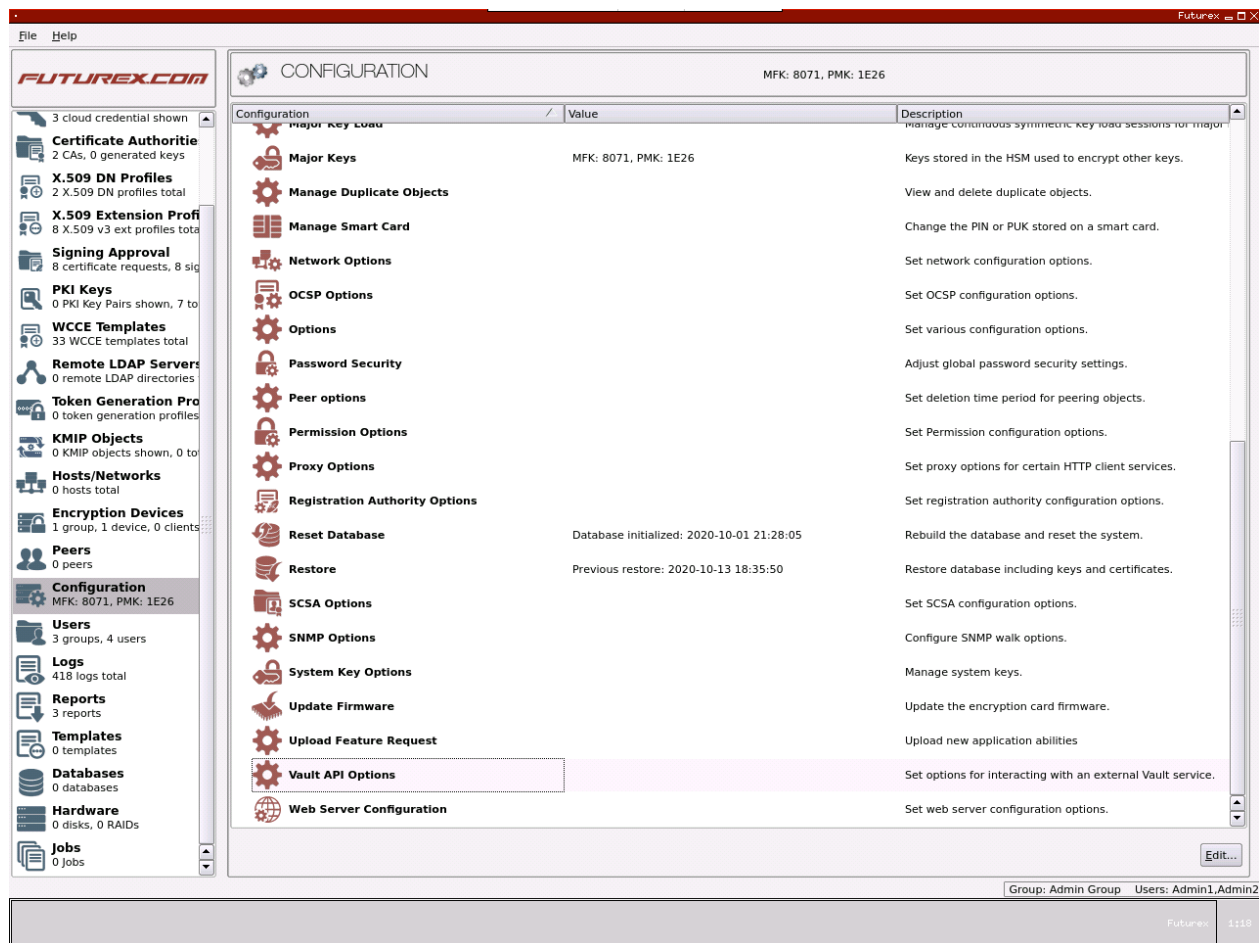
In the "Password" field, click the "Enter" button and set the same password that was set for the user created in Vault. Click "Save".

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

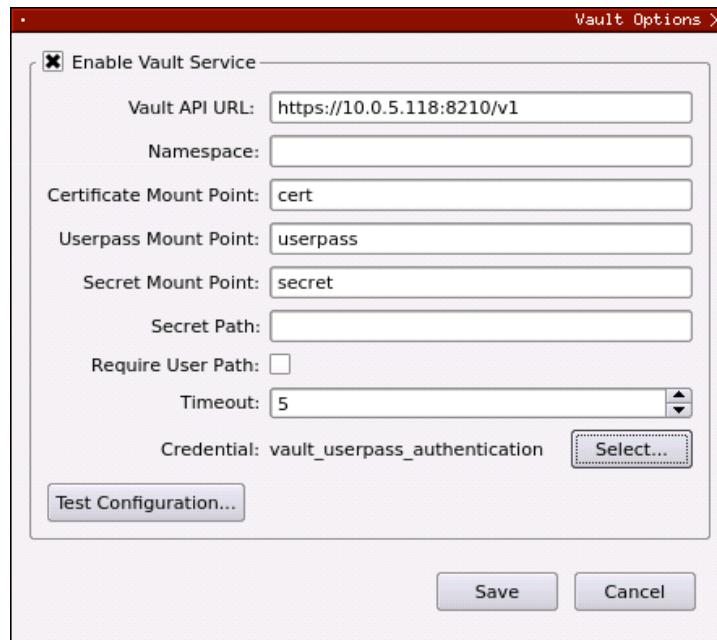


### Testing userpass authentication

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



Check the "Enable Vault Service" box.



Vault Options

Enable Vault Service

Vault API URL:

Namespace:

Certificate Mount Point:

Userpass Mount Point:

Secret Mount Point:

Secret Path:

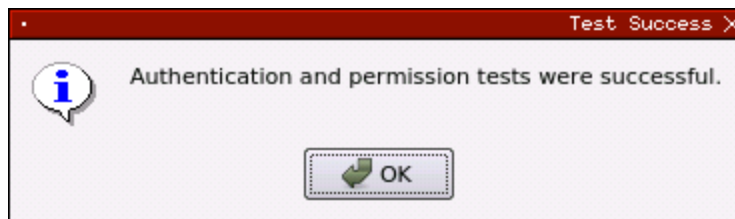
Require User Path:

Timeout:

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...".



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



## [4.2] TLS CERTIFICATES AUTH METHOD

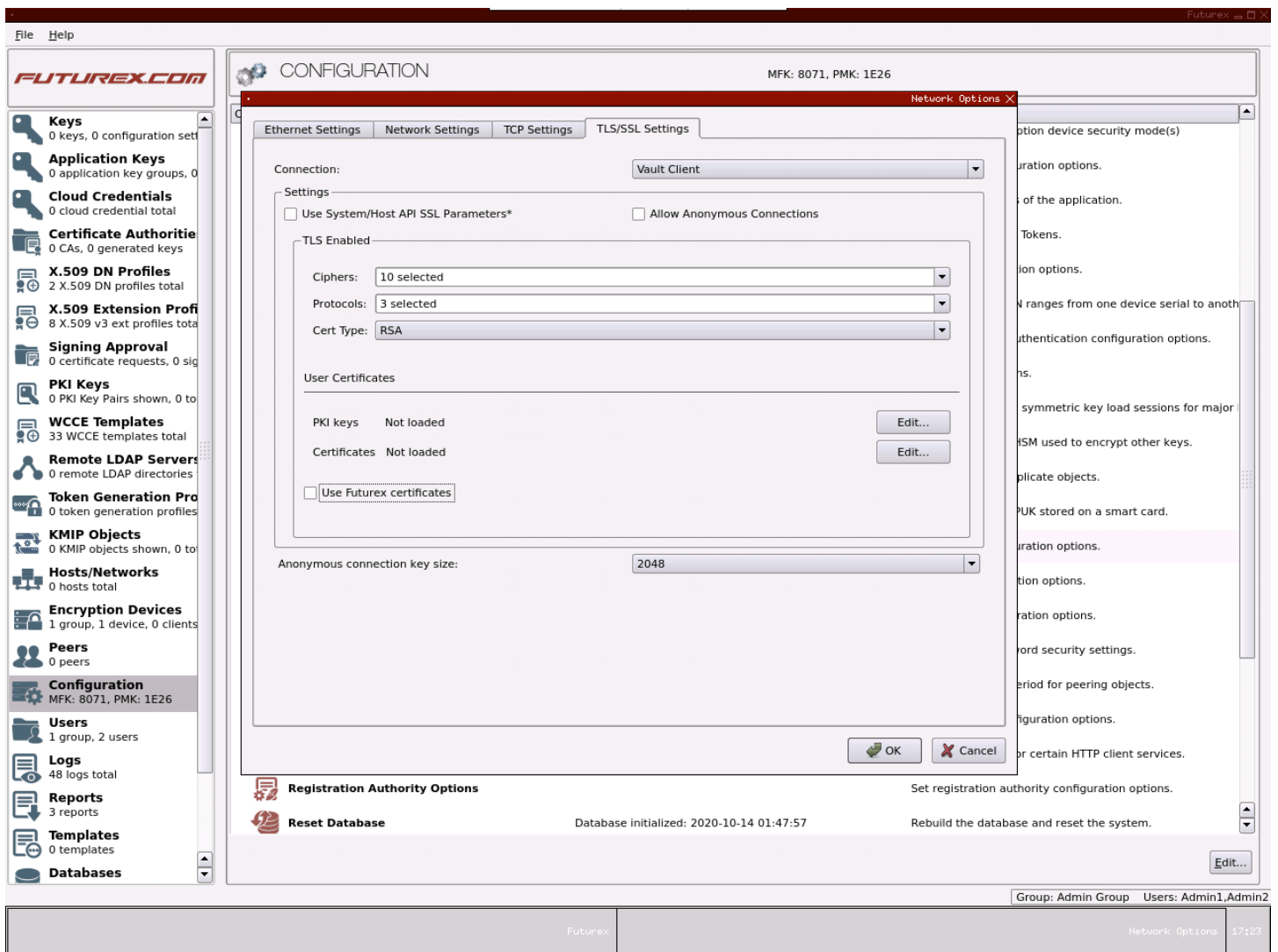
The **cert** auth method allows the KMES Series 3 to authenticate with Vault using SSL/TLS client certificates which are either signed by a CA or self-signed.

### Configure the Vault Client connection pair on the KMES Series 3

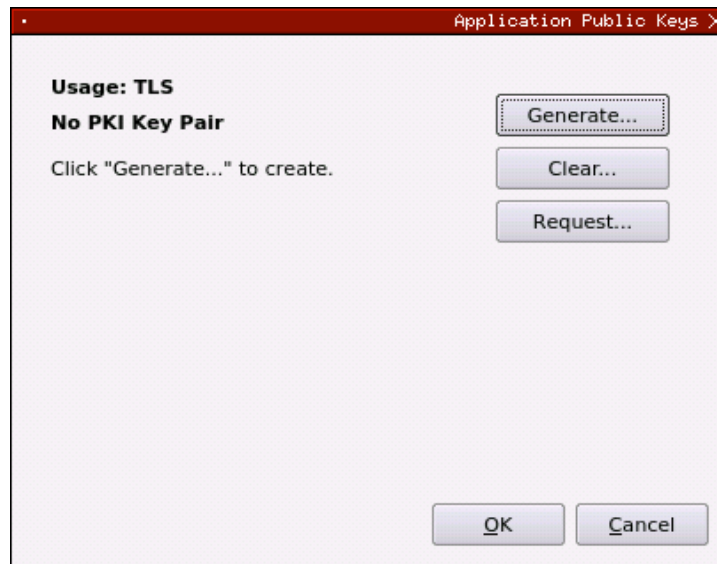
For the TLS Certificates auth method, it is necessary to first configure the **Vault Client** connection pair on the KMES Series 3 before configuring the **cert** authentication method in Vault.

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.

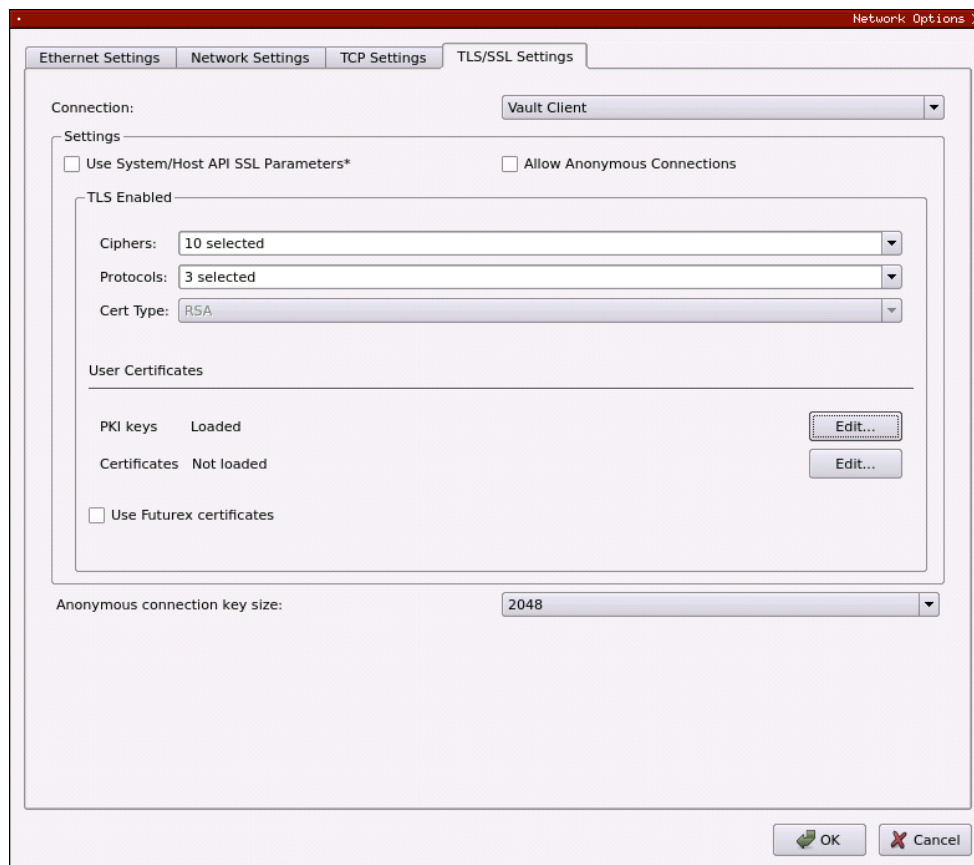


Uncheck the "Use System/Host API SSL Parameters" and "Use Futurex Certificates" boxes, then click the "Edit..." button next to "PKI keys" in the *User Certificates* section.

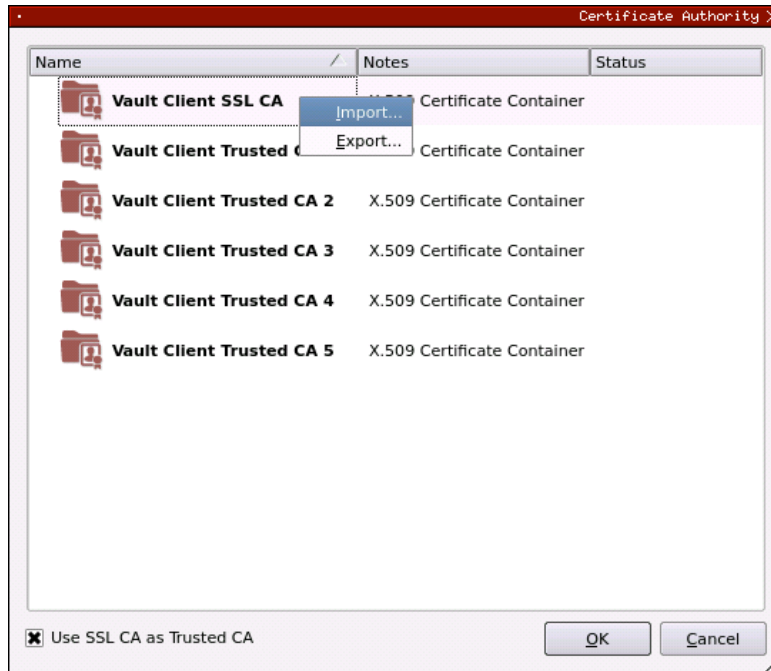


Generate a new PKI key pair, then request a CSR.

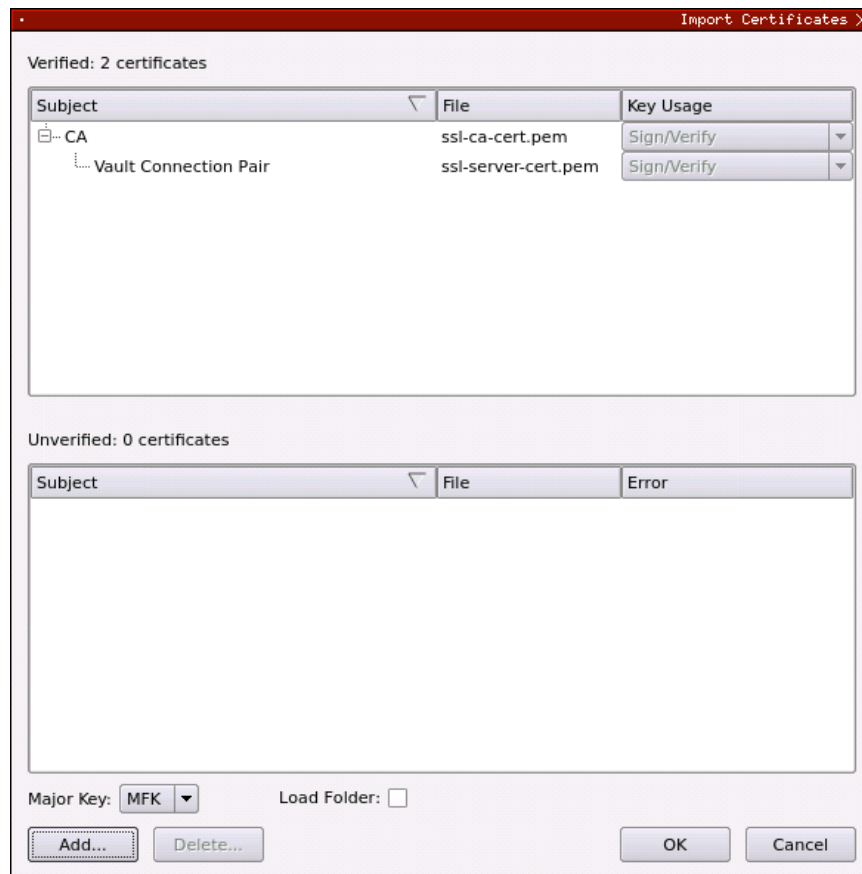
Next, the CSR just generated needs to be signed by a Certificate Authority (CA). For this demonstration, the CSR will be signed by the same CA that signed the client certificate that was configured in the Vault configuration file in section 3.2. Upload the CA certificate and the signed CSR to the storage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



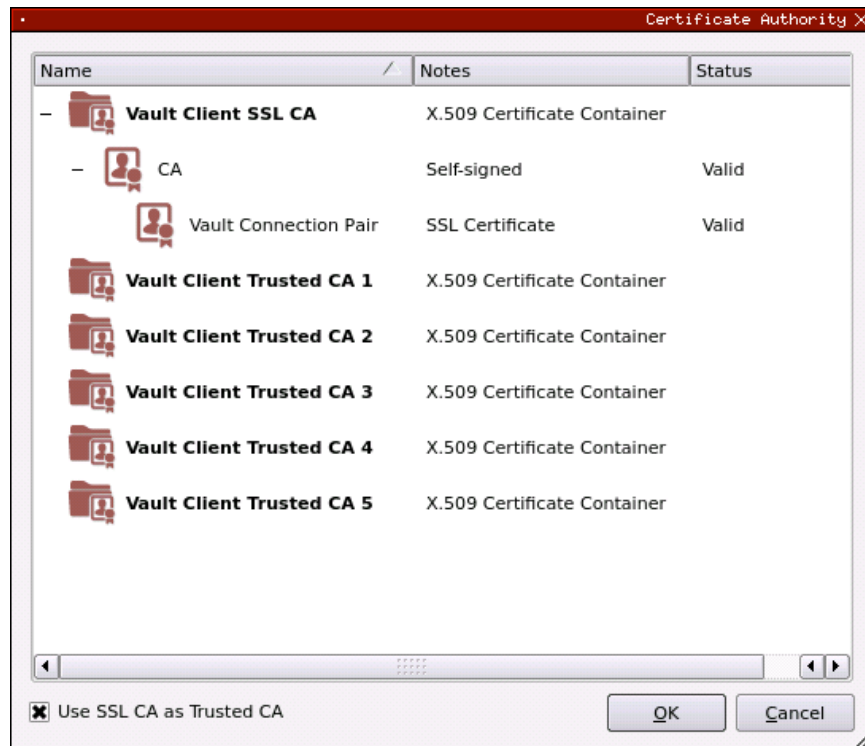
Right-click on the "Vault Client SSL CA" X.509 Certificate Container, then click "Import...".



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".



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



Click the "OK" button to save.

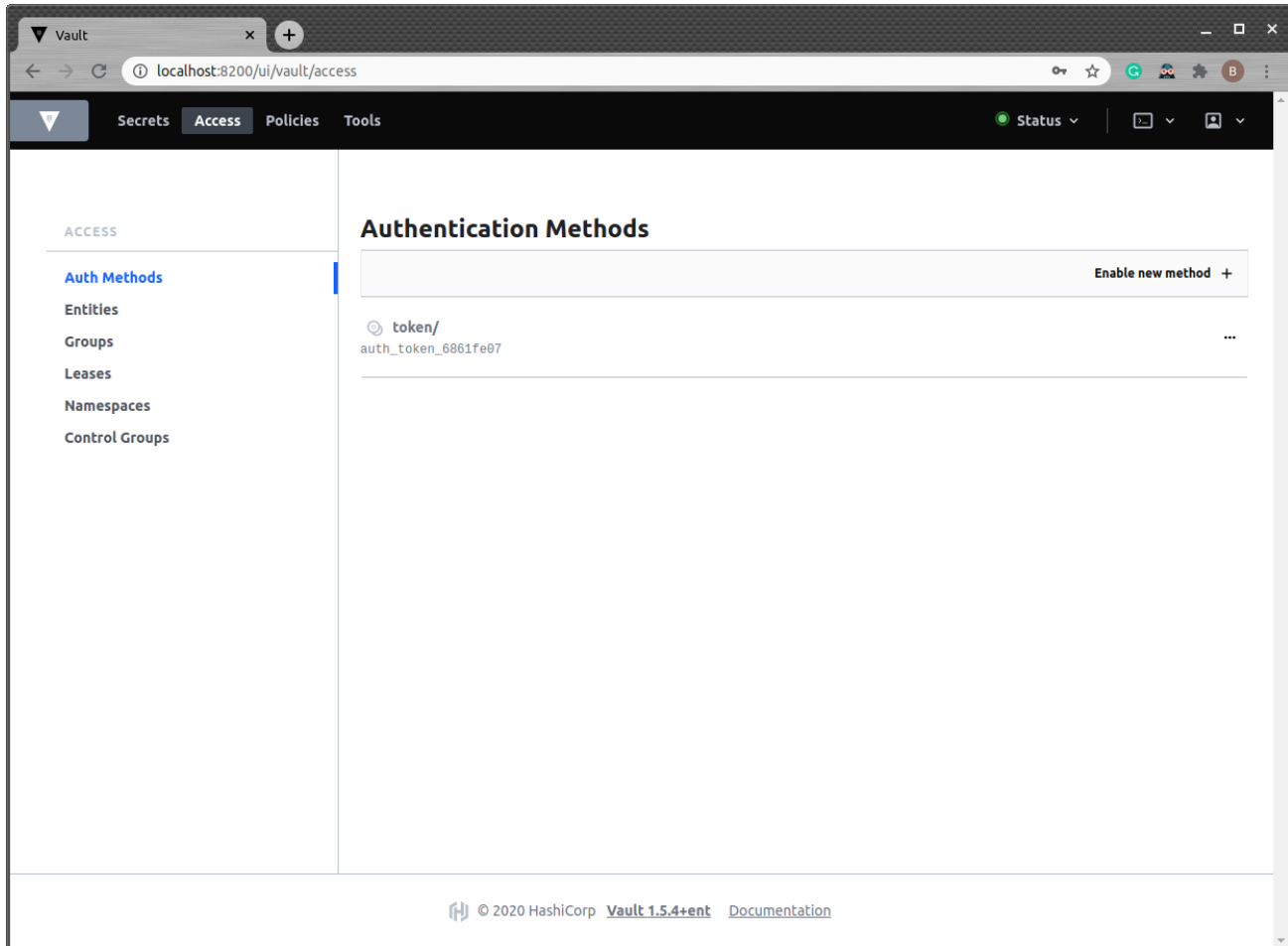
**NOTE:** As previously noted, for this demonstration, the same CSR was used to sign both the Vault Client connection pair CSR and the client certificate set in the Vault configuration file. If this were not the case, then the CA that signed the Vault Client connection pair CSR would need to be loaded 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.

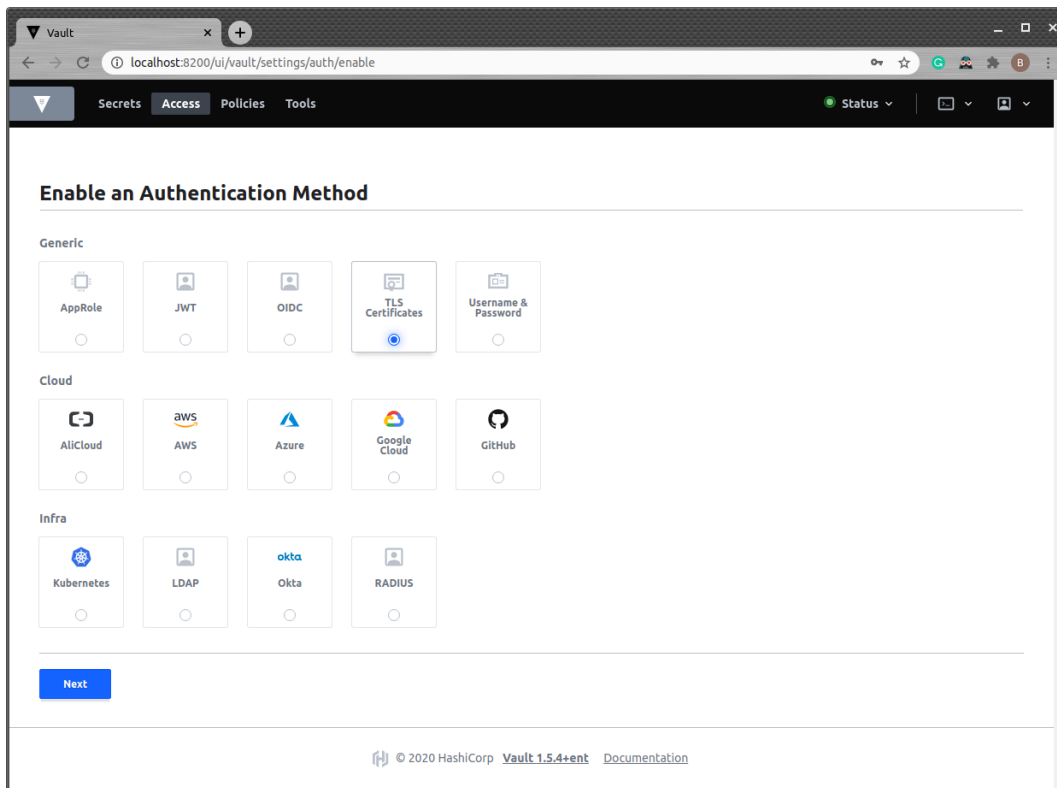
## Configuring cert authentication in Vault

### Method 1 - Using the Vault UI

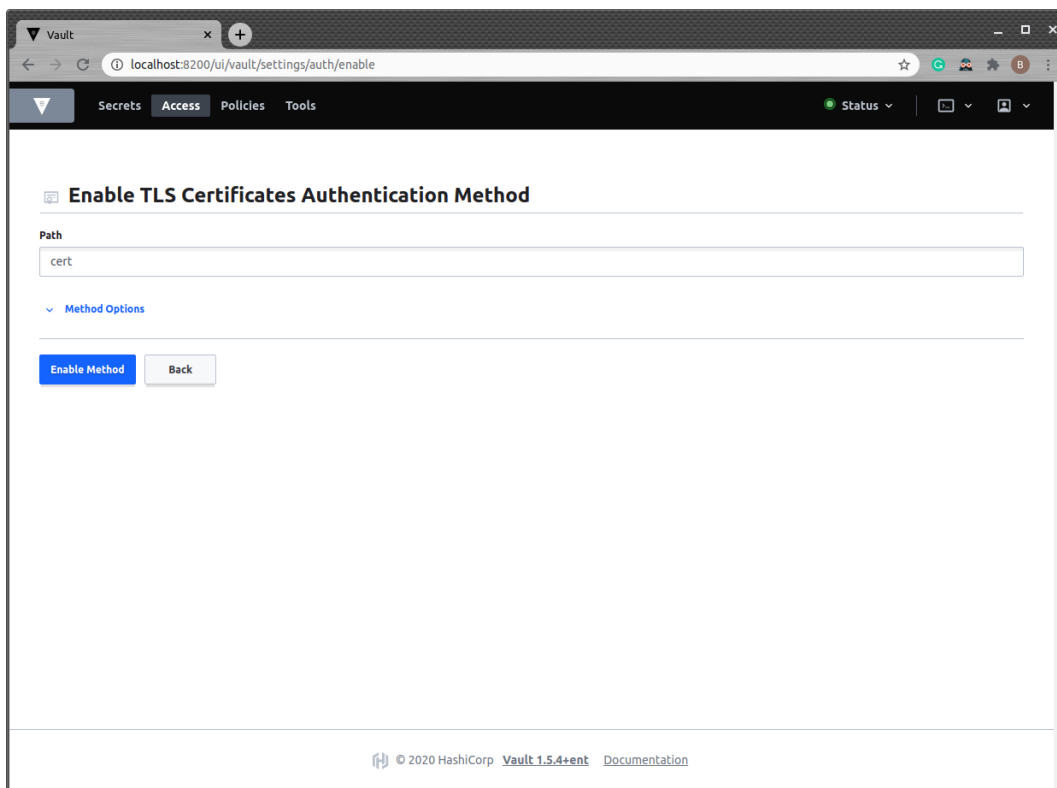
Navigate to the Access page in the Vault UI, then select "Enable new method".



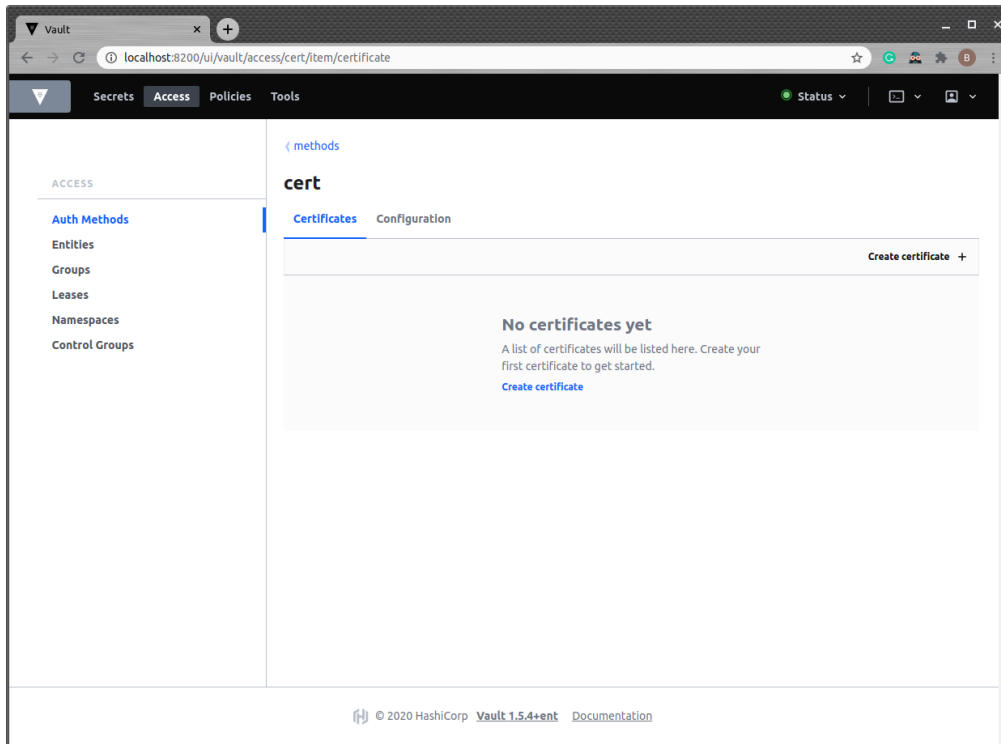
Select the "TLS Certificates" authentication method, then click "Next".



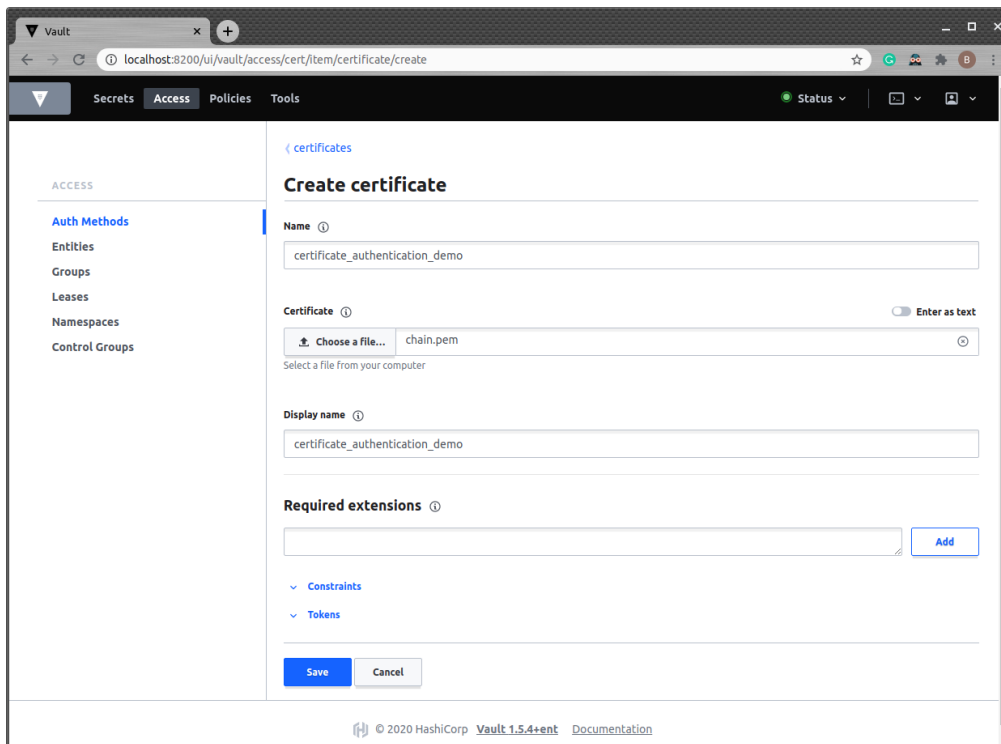
Leave the path as the default value, "cert", then click "Enable Method".



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
```

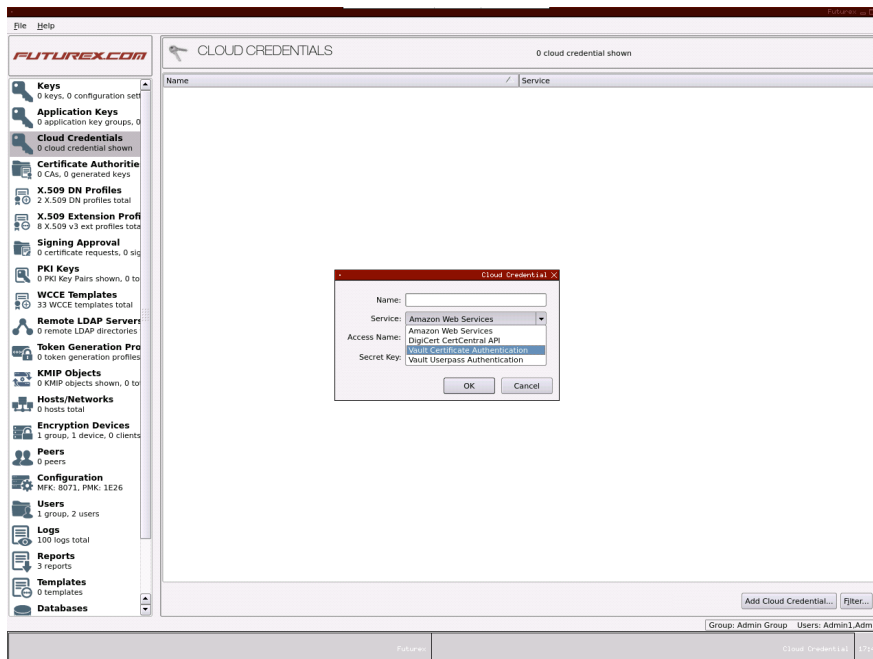
```
Success! Data written to: auth/cert/certs/certificate_authentication_demo
```

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.

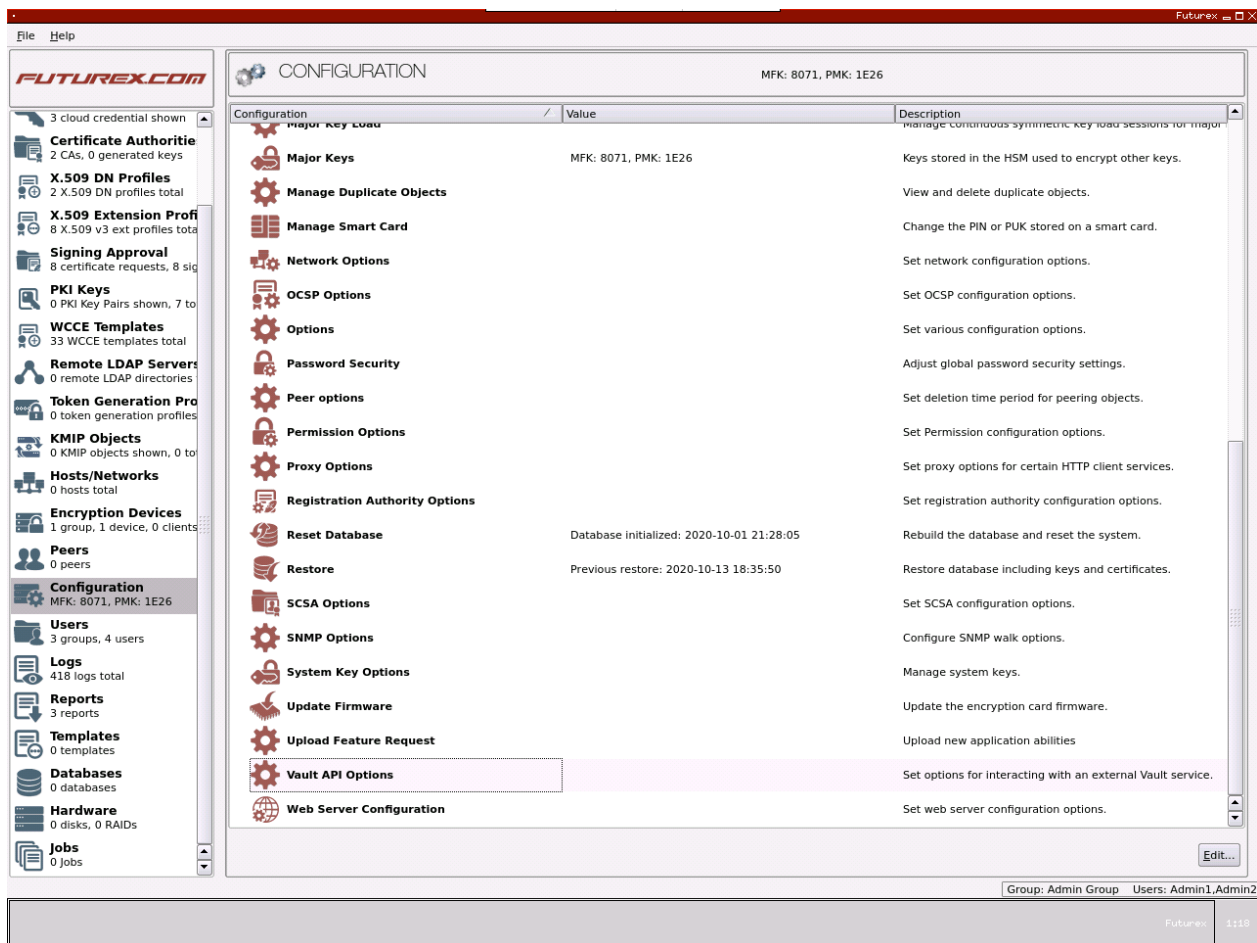
The value for the "TLS Config" field defaults to "Vault Client". This configures the Cloud Credential to use the "Vault Client" connection pair for authenticating with Vault.

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

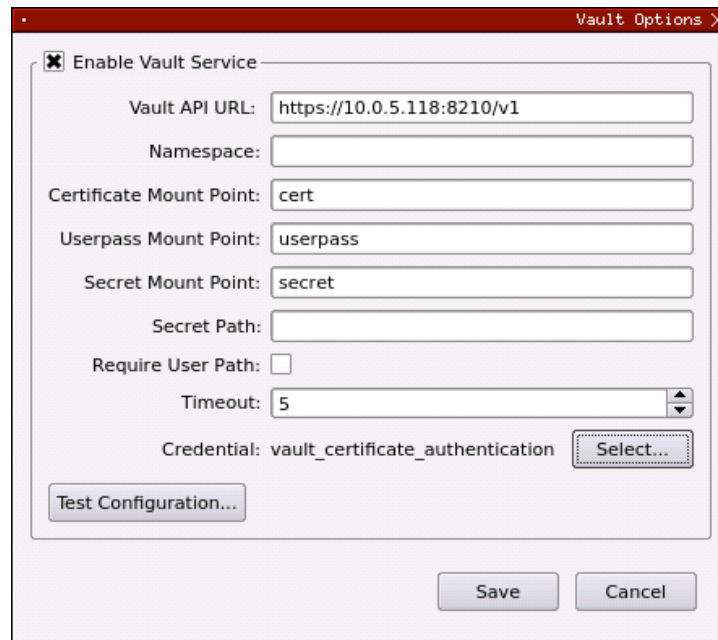


### Testing cert authentication

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



Check the "Enable Vault Service" box.



Vault Options

Enable Vault Service

Vault API URL:

Namespace:

Certificate Mount Point:

Userpass Mount Point:

Secret Mount Point:

Secret Path:

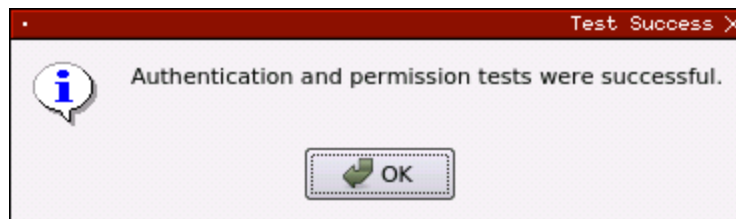
Require User Path:

Timeout:

Credential:

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...".



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

## [5] 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.

### [5.1] 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.

### [5.2] 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;]
```

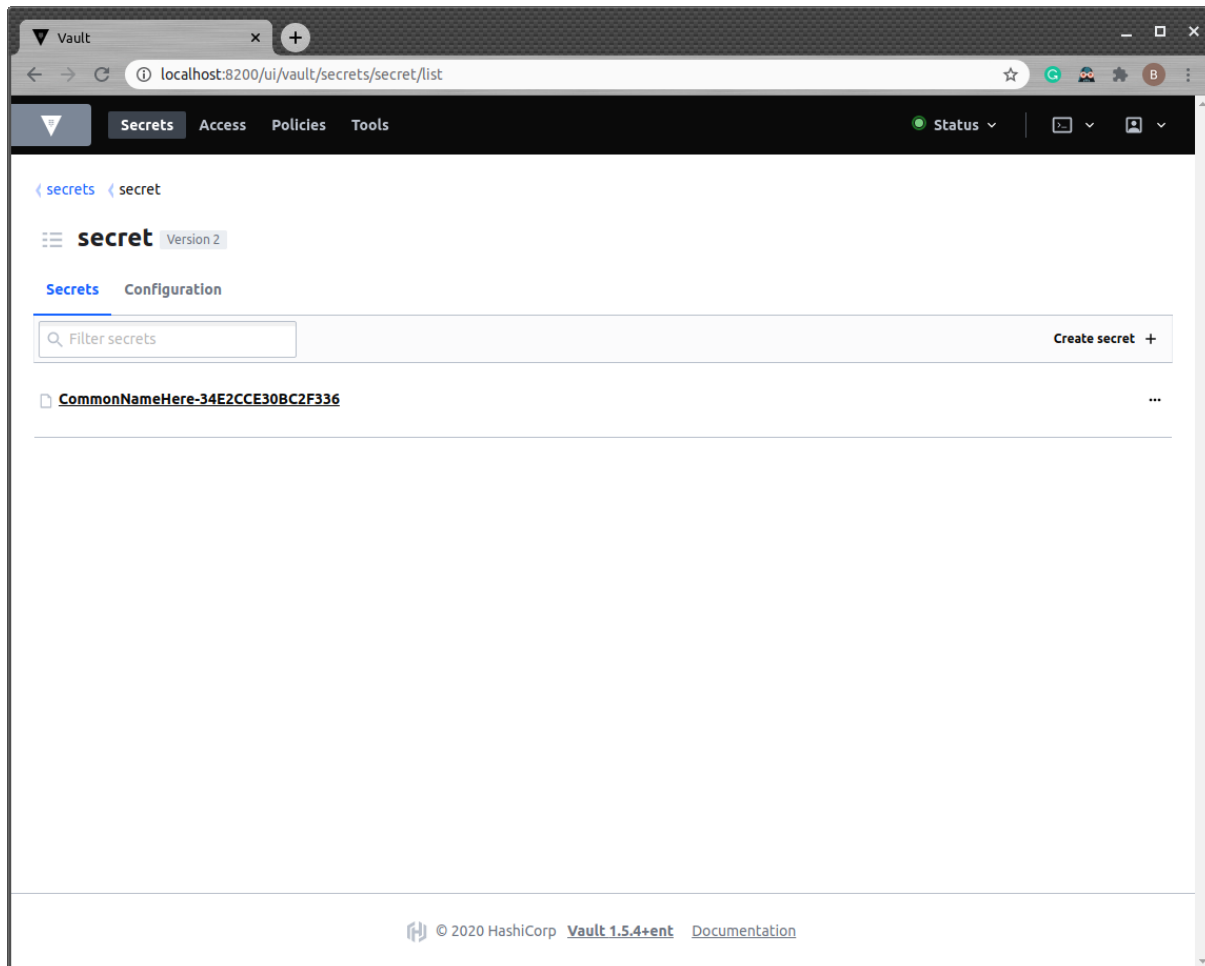
Now that we're logged in, we can run the RAUP command to upload a new X.509 PKI request.

```
[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;]
```

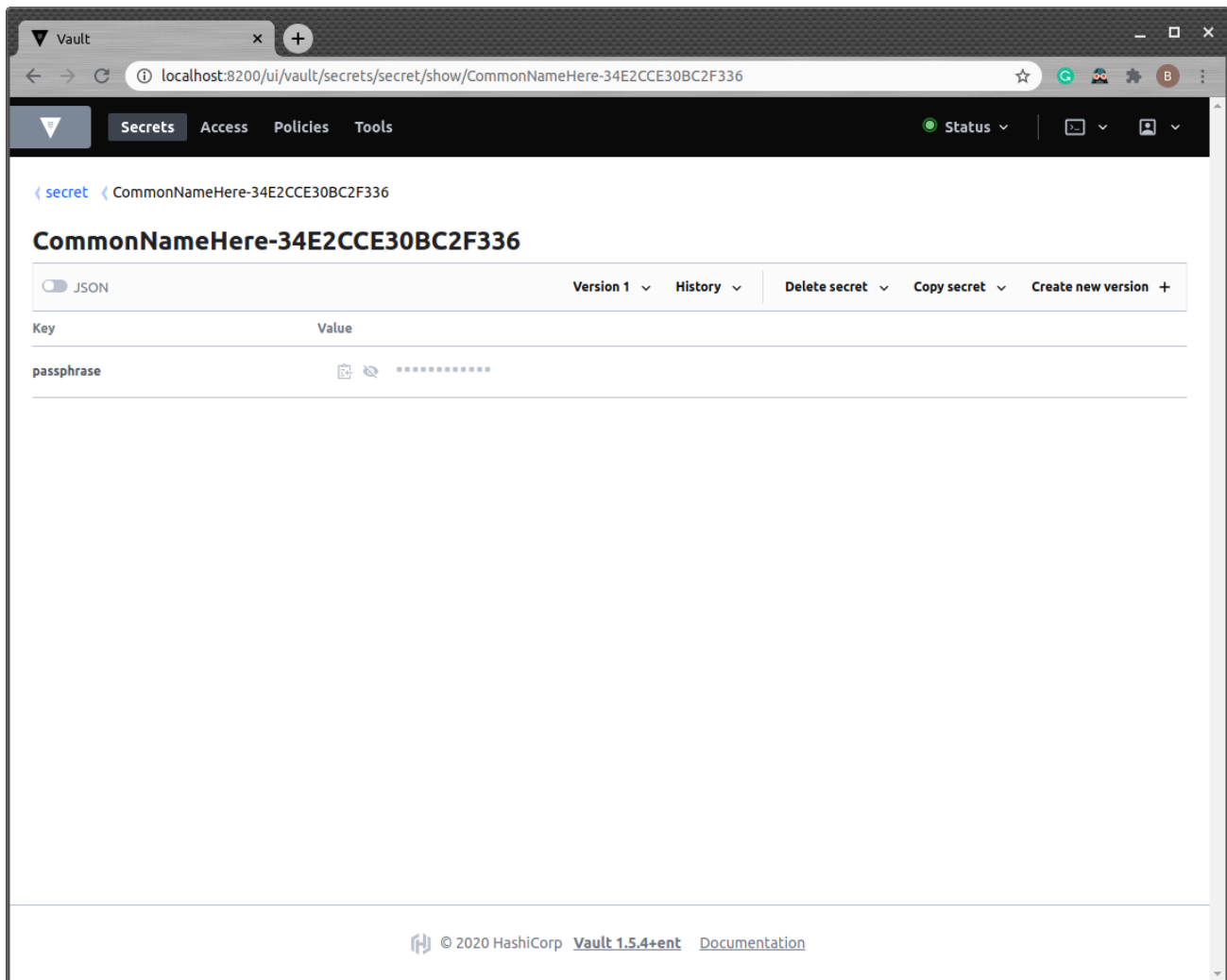
If the command succeeds, as it does above, a new X.509 PKI, issued as a PKCS #12 file, will be generated on the KMES Series 3, and the passphrase for the PKCS #12 file will be stored in Vault.

**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.



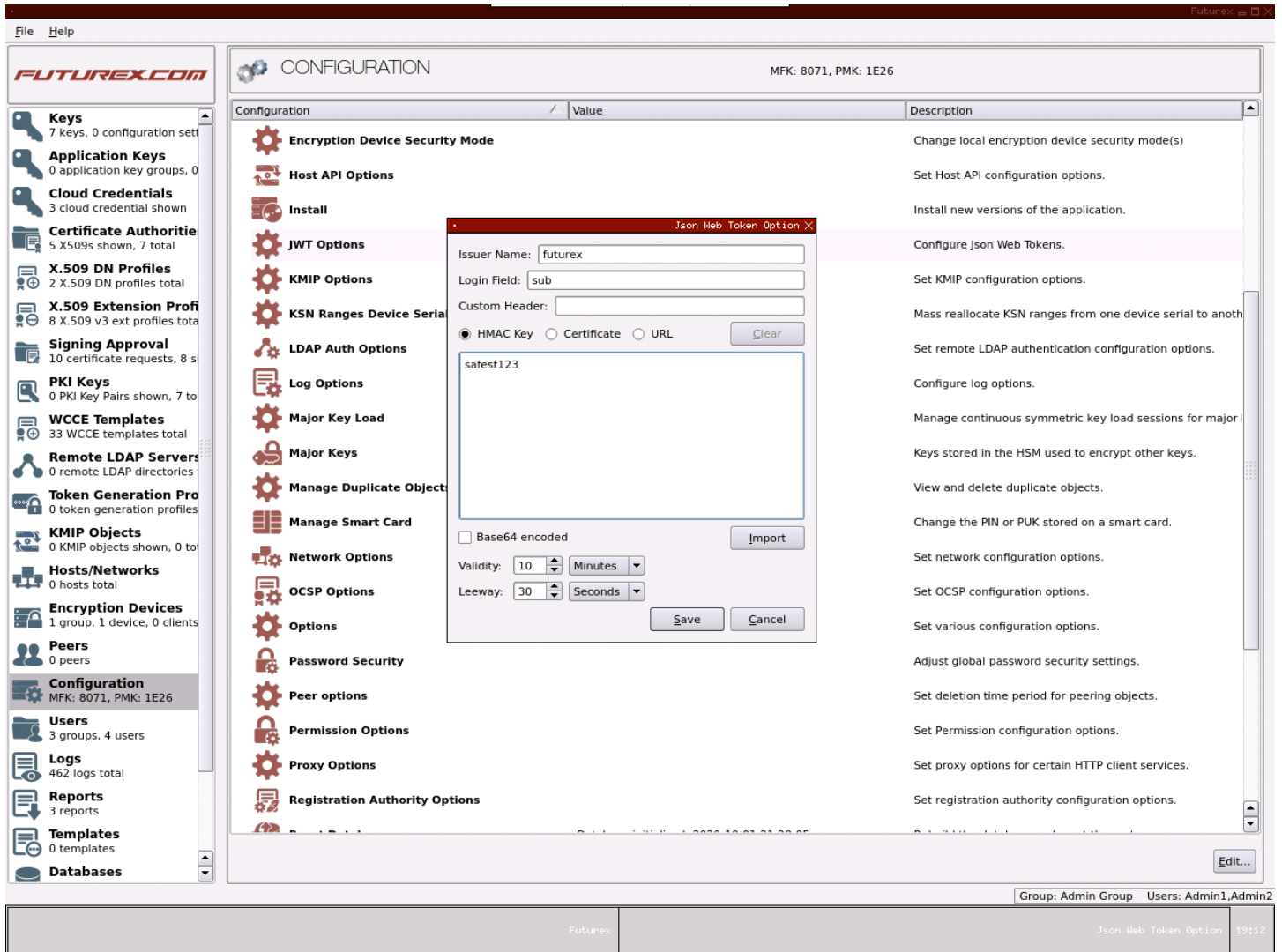
We see that the *Key* is "passphrase" and the *Value* is the passphrase for the PKCS #12 file that was generated on the KMES Series 3.

## [5.3] 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*.

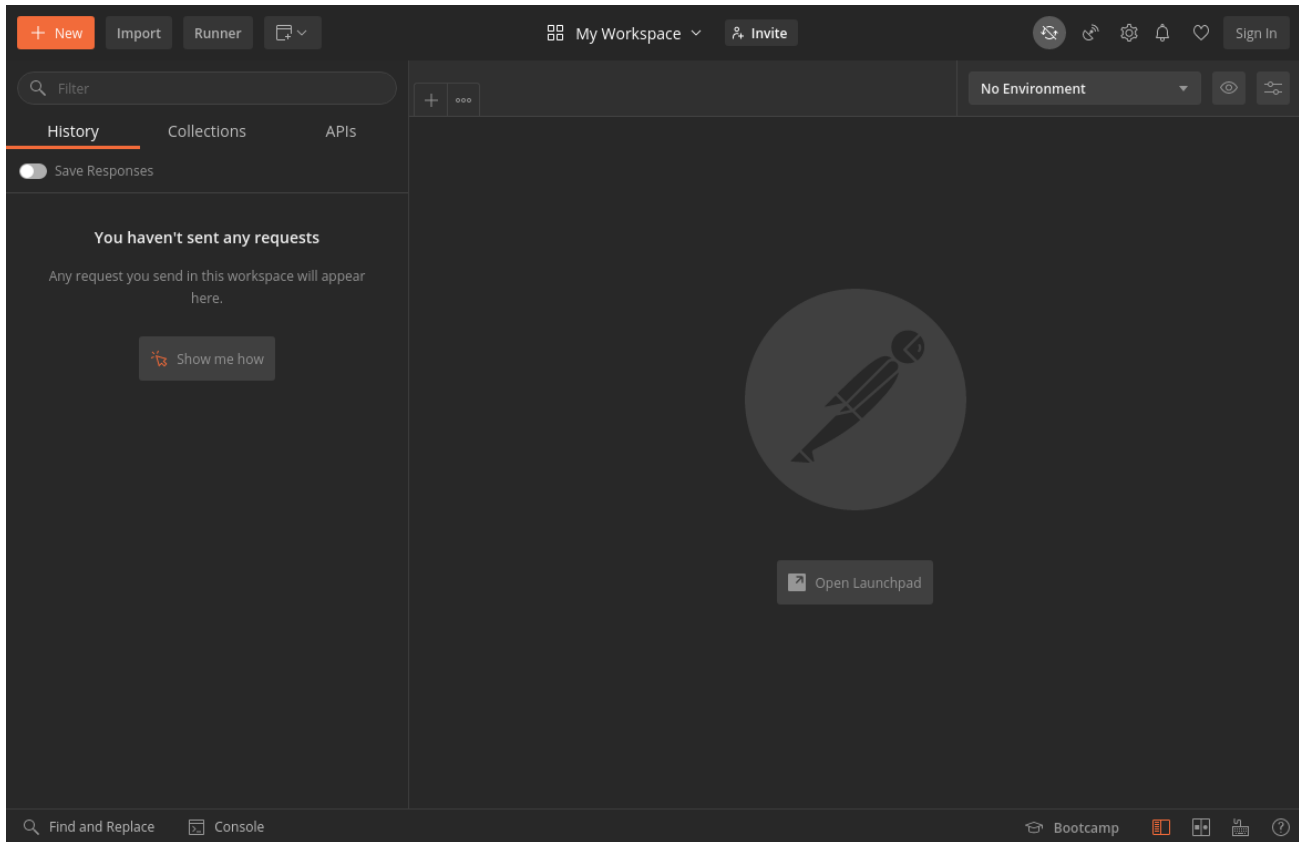


In the *Json Web Token Option* dialog, set the *Issuer Name* to "futurex" and set "safest123" as an HMAC Key password. The remaining fields can be left as their default values. Then, click "Save".

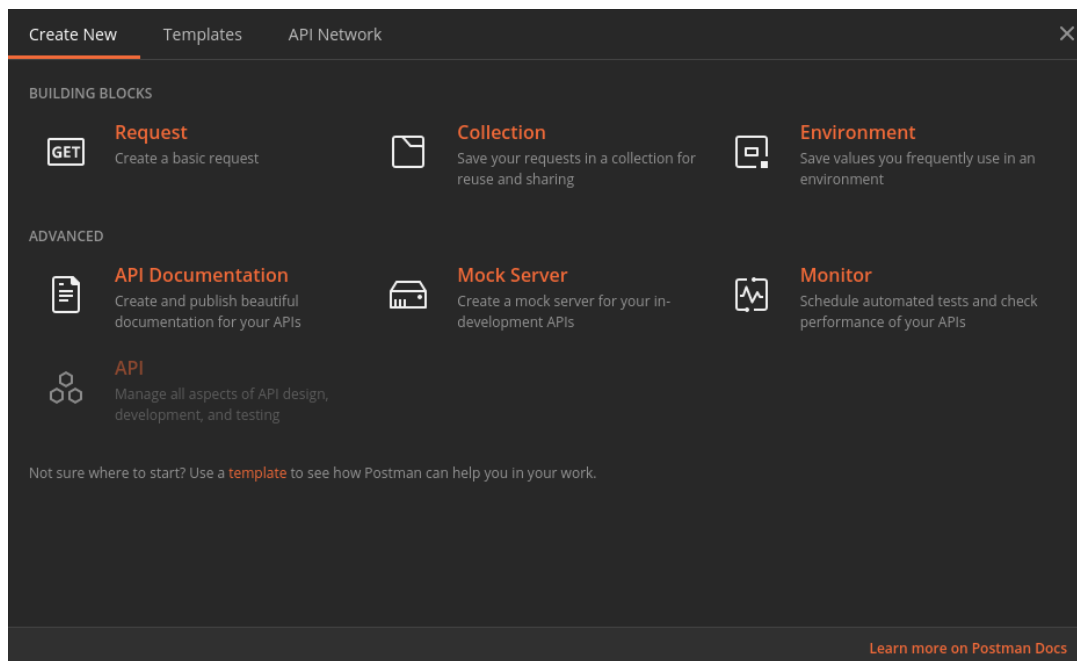
### Sending a POST request to the KMES RESTful API using Postman

**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.

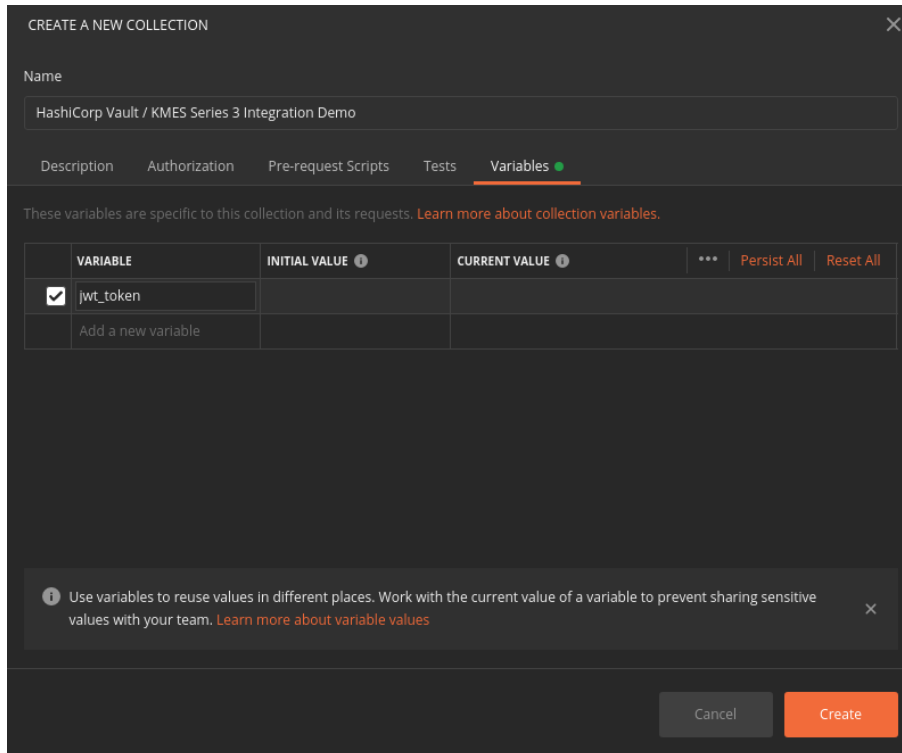
Start the Postman application, then click the orange "New" button in the top left area of the page.



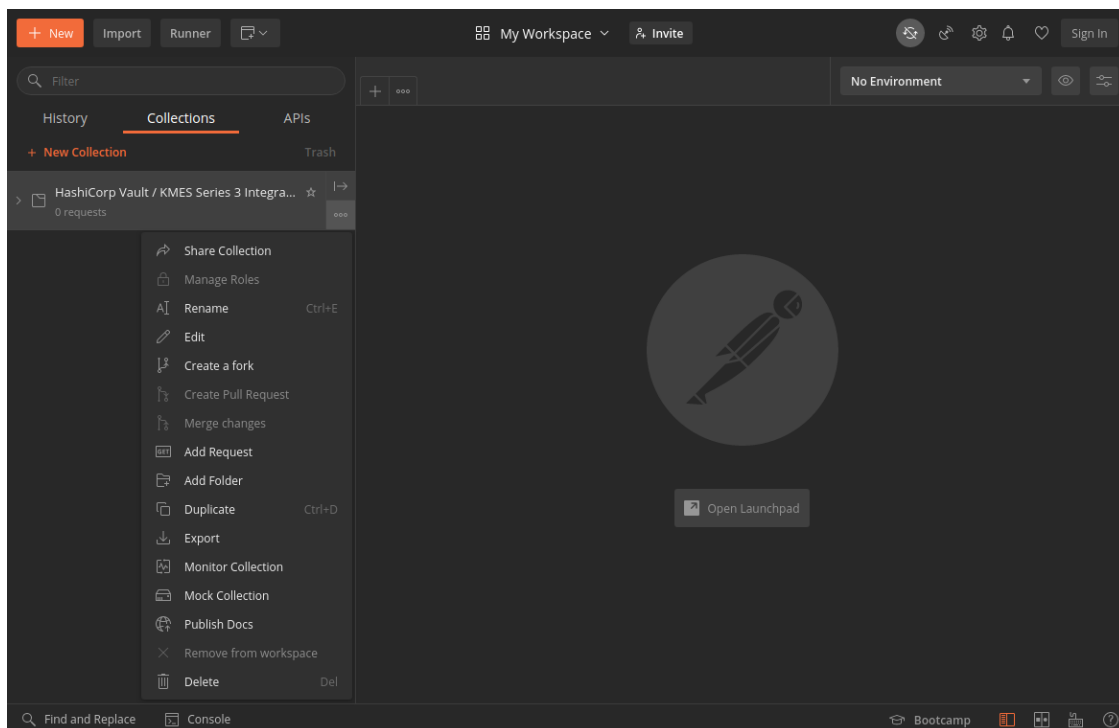
Select "Collection".



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).

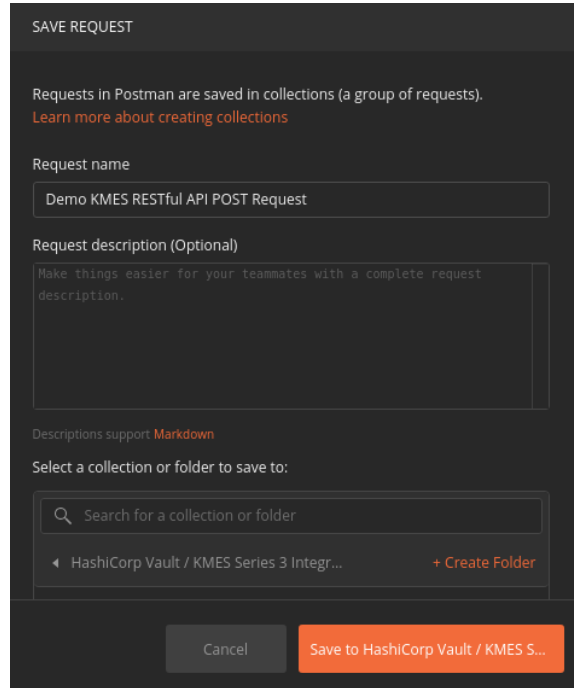


Right-click on the 3 dots in the new Collection and select "Add Request".

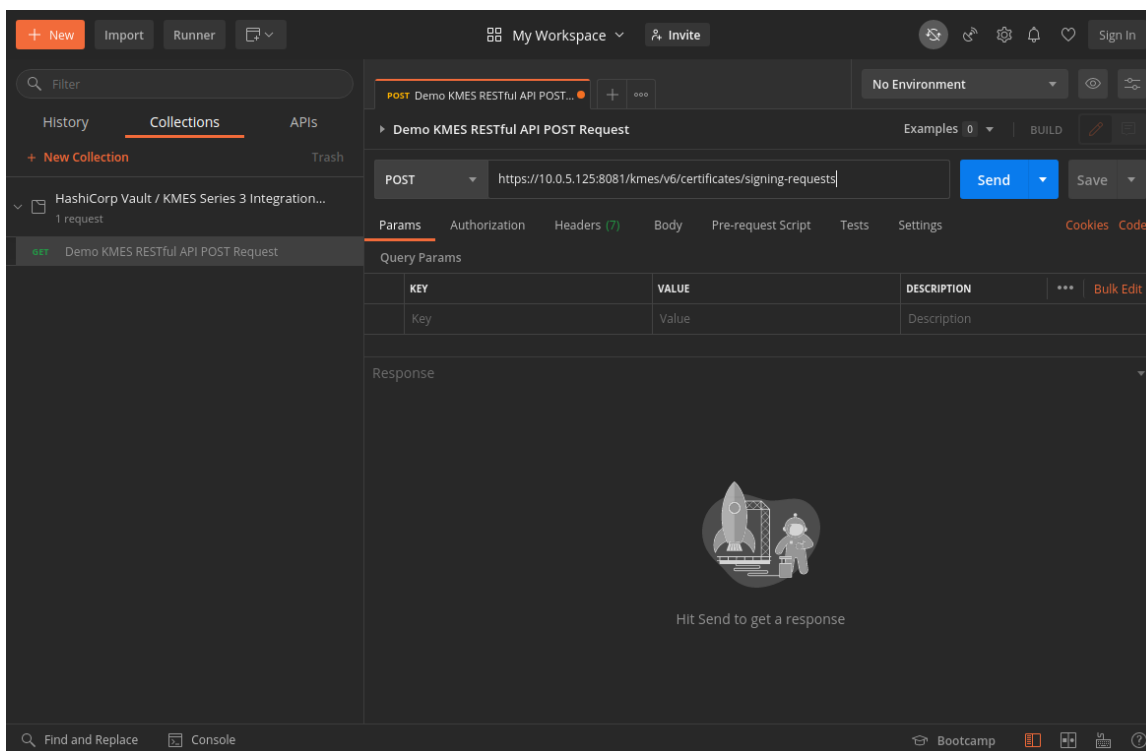




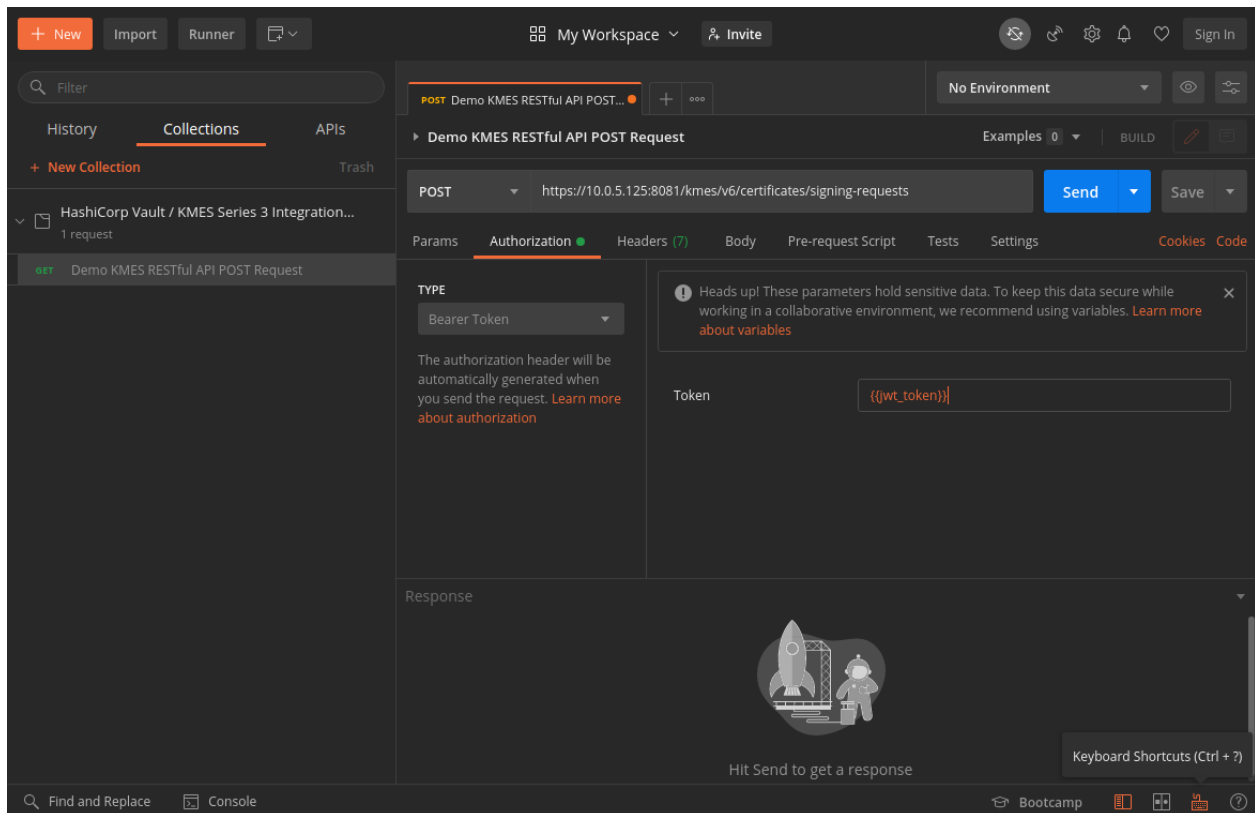
Set any name for the request. The most important part is that the request is associated with the newly created Collection. Save the request.



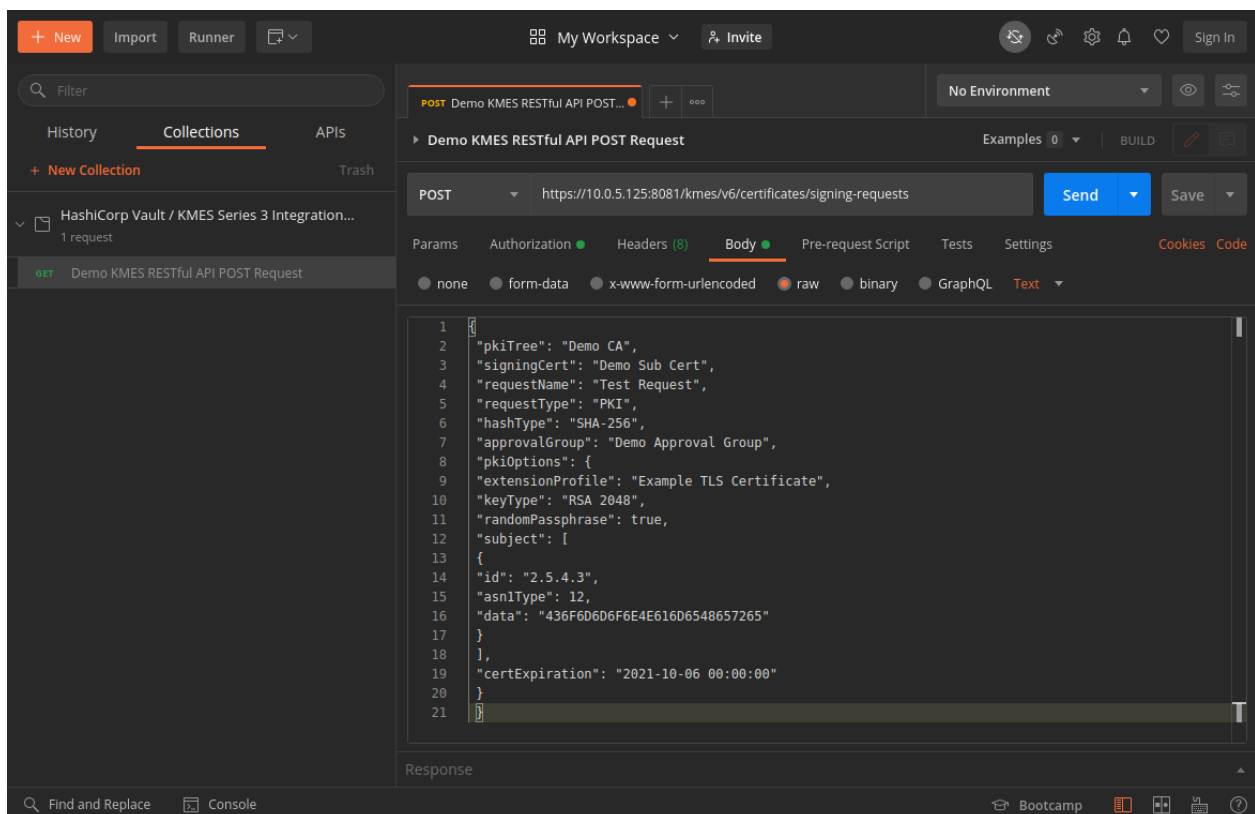
Expand the Collection folder and select the request that was just created. Set the request URL to "https://10.0.5.125:8081/kmes/v6/certificates/signing-requests" and change the request type to "POST" in the drop-down.



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

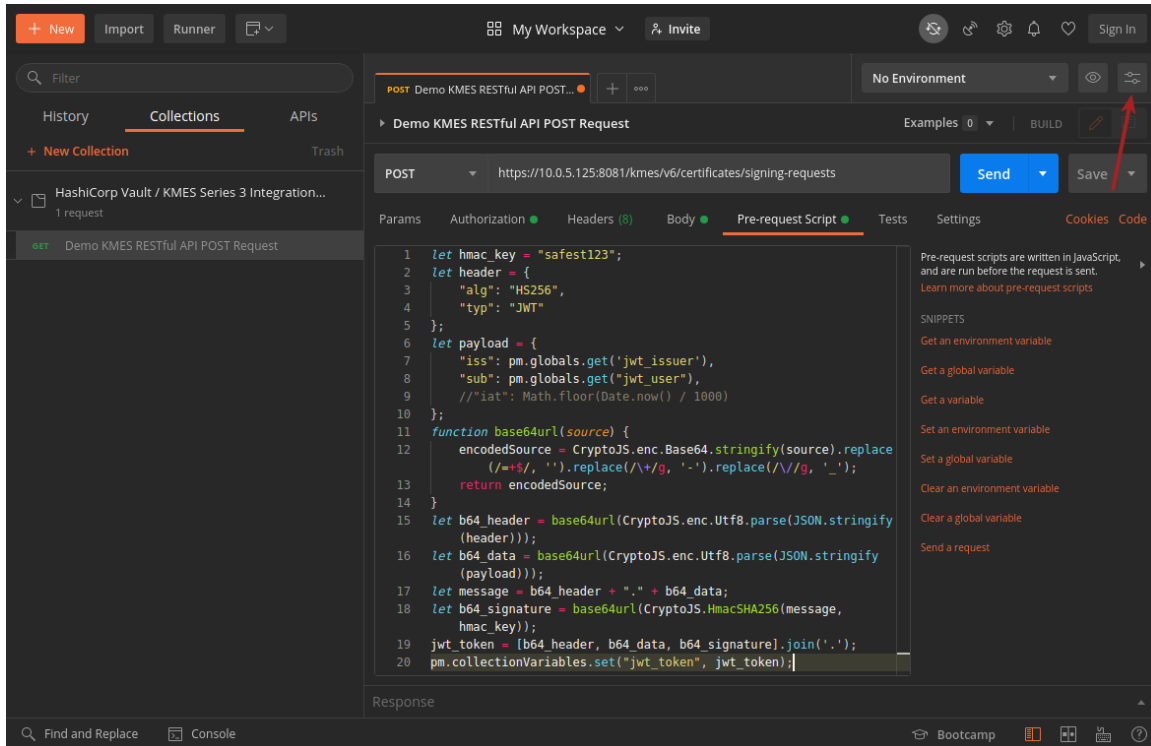


Navigate to the *Body* tab, select the "raw" bullet, then paste in the JSON shown below.

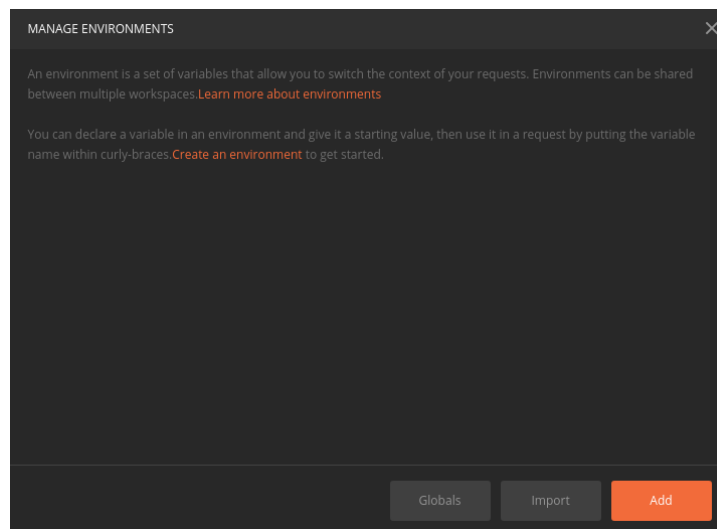


The request's body is where we specify that we want the KMES Series 3 to create a new X.509 PKI. The CA tree that we built on the KMES is defined, along with several other parameters. One of the most important parameters to notice is the "randomPassphrase" value set to "true". The "randomPassphrase"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

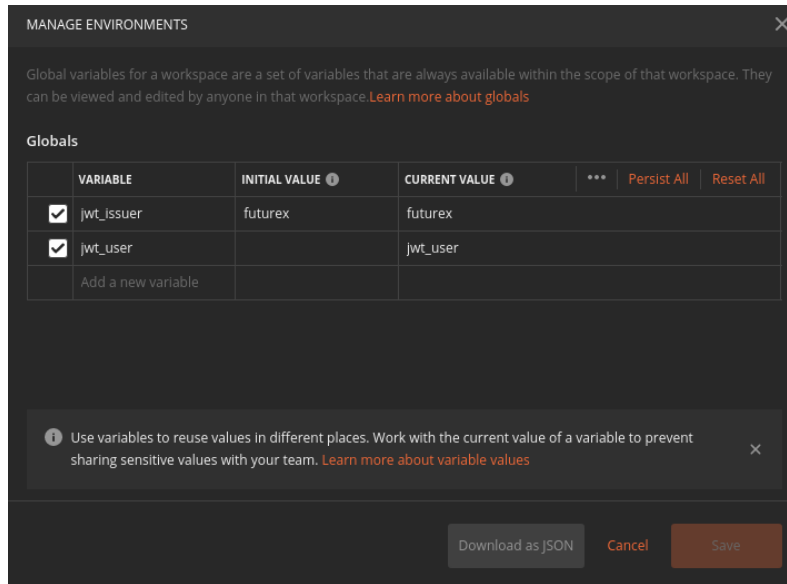
Navigate to the *Pre-request Script* tab and paste in the Javascript below. This code is responsible for generating the JWT token used for authentication to the KMES, on-the-fly. You'll notice that the value for "hmac\_key" in line 1 matches what was set in the JWT Options menu on the KMES.



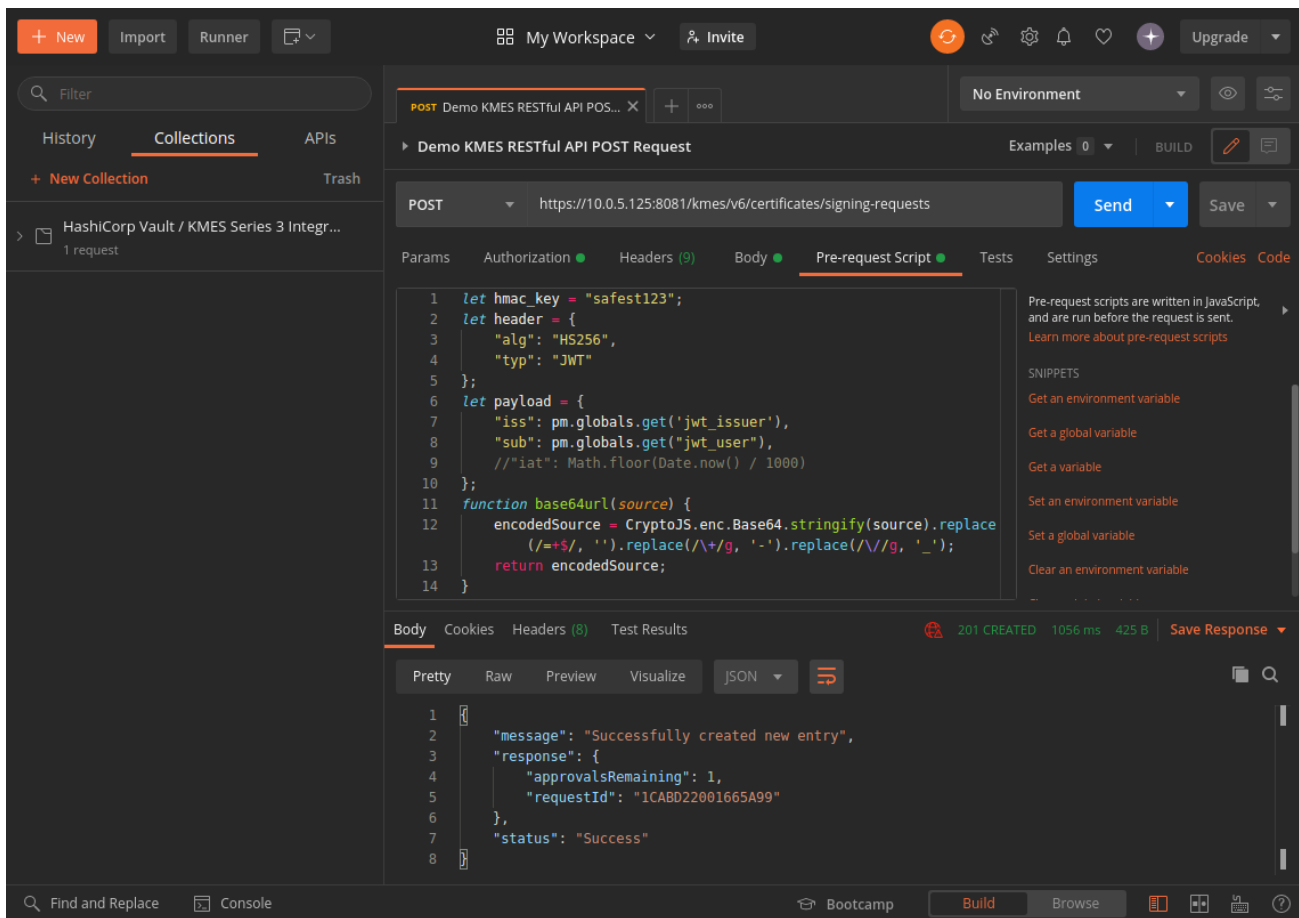
After pasting in the Javascript, click on the settings icon in the upper-right area of the page. Click the *Globals* button.



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

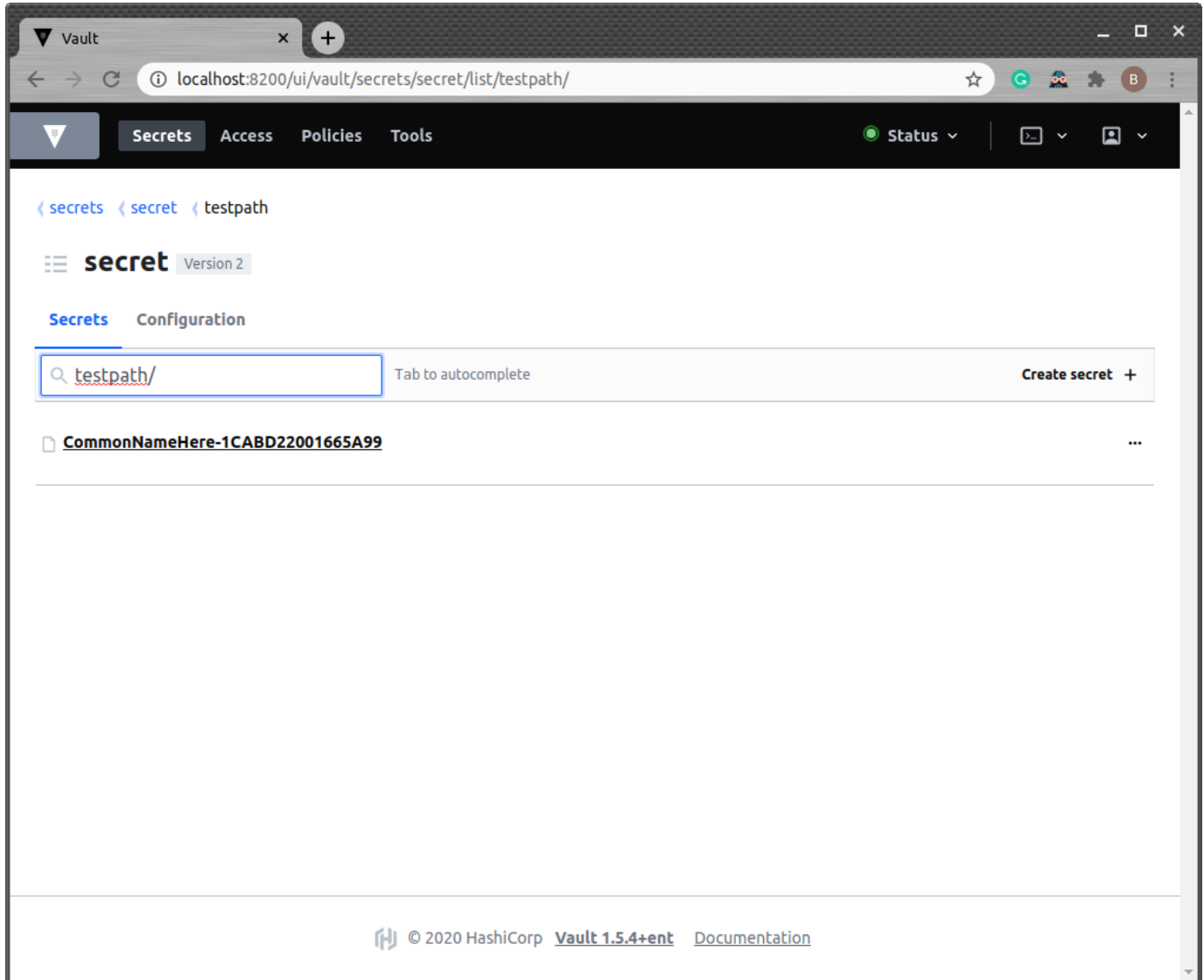


The setup is complete. Now click "Save" and then send the POST request.



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](mailto: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](mailto:info@futurex.com)

#### EXCEPTIONAL SUPPORT

24x7x365

Toll-Free: 1-800-251-5112

E-mail: [support@futurex.com](mailto:support@futurex.com)

#### SOLUTIONS ARCHITECT

E-mail: [solutions@futurex.com](mailto:solutions@futurex.com)