SECTIGO®
HashiCorp Vault Integration Guide

Version 1.0

December 2019

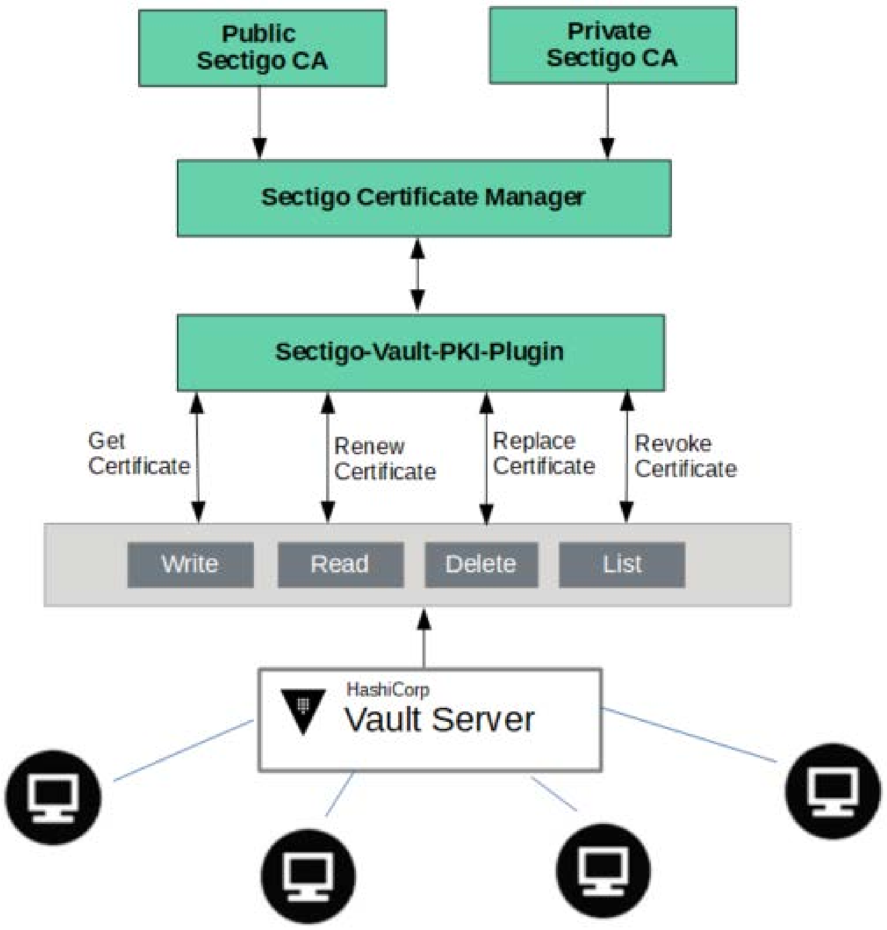
Sectigo HashiCorp Vault Integration

The Sectigo HashiCorp Vault integration provides a seamless solution for the enrollment, collection, revocation, renewal, and replacement of SSL/TLS and client (S/MIME) certificates issued by the Sectigo Certificate Manager (SCM). This integration is distributed as a custom HashiCorp Vault (Vault) PKI plugin. It provides the following features:

- RSA 2048, 3072, and 4096-bit private key generation
- Certificate Signing Request (CSR)
- Storage and state tracking of certificates issued by SCM in Vault

The Sectigo Vault integration supports both the generation and storage of new SSL and client certificates in Vault. The Sectigo Vault PKI plugin supports verifying the validity of certificates that are being read from Vault; certificates that fall within a user-specified certificate expiry window may (optionally) get automatically renewed. There are various types of SSL and client certificates that can be requested by supplying the appropriate configuration options.

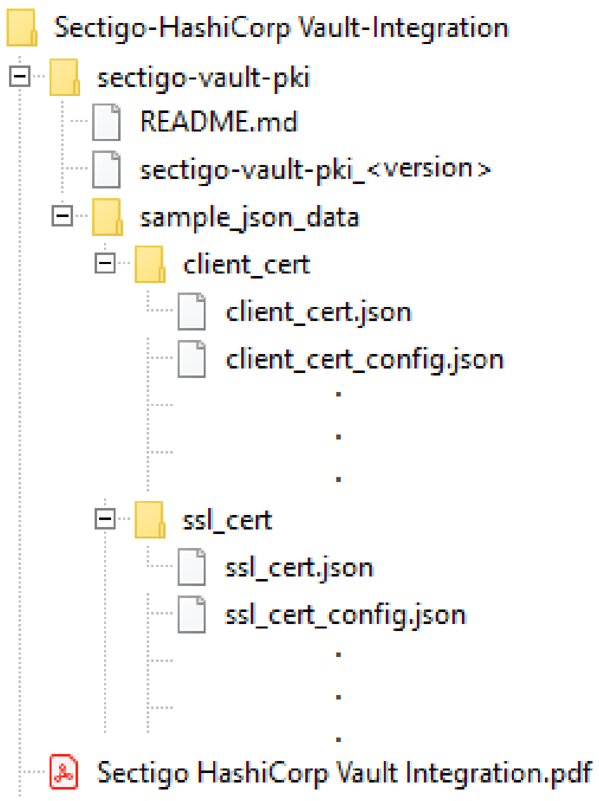
Note: The types of SSL/TLS and client certificates available to you are based on your account setup.



This integration guide describes the following:

1 Package Contents

- **sectigo-vault-pki**:
  - **sectigo-vault-pki_<version> (binary)**: The Sectigo Vault PKI plugin that allows users to store and manager certificates that get generated from SCM on Vault.
  - **sample_data_jsons**: Sample JSON files that can be used by the user to interact with the Sectigo Vault PKI plugin.
  - **README.md**: A README file that includes example commands that showcase how to use the Sectigo Vault PKI plugin.
- **Sectigo HashiCorp Vault Integration.pdf**: The integration user guide.



1.1 Prerequisites

- HashiCorp Vault Version 1.2.3 or higher[1]
- jq (optional)
- shasum or equivalent
- SCM organization with Web API access enabled for both SSL and client certificates
- List of SSL and client types with associated terms for the organization
- Supported operating systems:
    - Linux operating systems:
        - Ubuntu Server 16.04 LTS
        - Ubuntu Server 18.04 LTS
        - Ubuntu Server 19.04
        - Ubuntu Server 19.10
        - CentOS 7.3

2 Understanding the Integration

Backend plugins in Vault are essentially separate, standalone applications that Vault executes and communicates with over RPC. Each backend plugin acts as a server and exposes certain API endpoints, which Vault would then interact with.

The Sectigo Vault PKI plugin is a custom secrets backend plugin which makes use of the Sectigo Go library in order to send HTTP requests to the SCM APIs. The plugin exposes its own API endpoints which each correspond to a specific path that builds on top of a starting base path prefix. For the purpose of this document, the path prefix `sectigo-vault-pki` is used.

Note: You can rename this path prefix when you mount the Sectigo Vault PKI plugin into your Vault server.

2.1 Components

The Sectigo HashiCorp Vault integration is based on the following two components:

- **Go client library for the Sectigo API** – handles the communication with the Sectigo REST API.
- **Sectigo Vault PKI Plugin** – integrates with Vault and mediates the interaction between the user, Vault, and the Sectigo REST API.

[1] Vault version 1.2.3 has a known issue with reading the VAULT_CLIENT_TIMEOUT environment variable when executed through the Vault CLI tool. This issue was rectified in Vault 1.3.0. For more information read the Dealing with Timeouts and Error Codes section.

2.2 Path Endpoints

The Sectigo Vault PKI plugin exposes several paths that users can interact with. Different paths are defined for different use cases. Depending on the applicable functionality, each path accepts different input parameters.

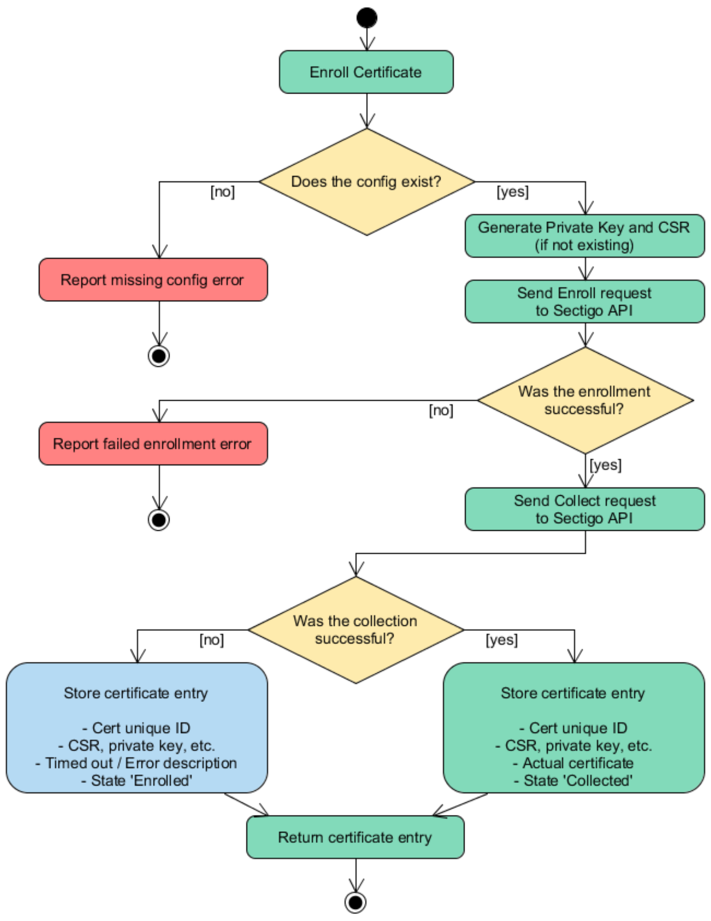
The following table lists all the paths that are supported by the Sectigo Vault PKI plugin and displays the operations that are supported by each path[2]:

Path	Operation			
	Write	Read	List	Delete
configs	✓	✓	✓	✓
enroll	✓			
certs		✓	✓	✓
revoke	✓			
replace	✓			
renew	✓			
fetch	✓			

You can retrieve in-code API help for each individual path in Vault by using the built-in `path-help` CLI command. For more information on `path-help`, see https://www.vaultproject.io/docs/commands/path-help.html.

In order to enroll and manage certificates on Vault through SCM, users must first create a `config` entry in Vault; a config entry may be used for enrolling and managing multiple certificates that correspond to the same SCM configuration. The following diagram illustrates a typical certificate enrollment scenario using the Sectigo Vault PKI plugin:

[2] Sample Vault CLI commands can be found in the Interacting with the Plugin section.



Enroll Certificate
Does the config exist?
[no]
[yes]
Report missing config error
Generate Private Key and CSR (if not existing)
Send Enroll request to Sectigo API
Was the enrollment successful?
[no]
Report failed enrollment error
[yes]
Send Collect request to Sectigo API
Was the collection successful?
[no]
[yes]
Store certificate entry - Cert unique ID - CSR, private key, etc. - Timed out / Error description - State 'Enrolled'
Store certificate entry - Cert unique ID - CSR, private key, etc. - Actual certificate - State 'Collected'
Return certificate entry

3 Understanding the Configurations

3.1 Parameter Descriptions:

The Sectigo Vault PKI plugin can handle a multitude of parameters. The parameters required are dependent on the applicable use case and they can be passed to Vault in two principal ways:

- Bundled together in a JSON file when interacting with Vault through a Vault CLI tool or through cURL commands.
- Individually as a key/value pairs when interacting with Vault through the Vault CLI tool.

3.1.1 Configuration and User Specific Parameters

Parameter	Type	Description
sectigo_cm_user	Mandatory	User ID to access your URI.
sectigo_cm_password	Mandatory	Password to access your URI.

sectigo_config_type	Mandatory	The configuration type for your Vault-specific config entry. This can either be `ssl_cert` or `client_cert`.
sectigo_cm_org_id	Mandatory	The Organization ID (numeric).
sectigo_cm_base_url	Mandatory	The base URL of the Sectigo Certificate Authority.

3.1.2 CSR Parameters

Parameter	Type	Description
sectigo_csr_domain	Conditional	Single value for a domain which is included in the certificate **Common Name (CN)** field. Required if `sectigo_csr` is not provided.
sectigo_csr_country	Conditional	The country name which is included in the certificate **Country (C)** field. Required if `sectigo_csr` is not provided.
sectigo_csr_state	Conditional	The state/private name which is included in the certificate **State (ST)** field. Required if `sectigo_csr` is not provided.
sectigo_csr_location	Conditional	The location name which is included in the certificate **Location (L)** field. Required if `sectigo_csr` is not defined.
sectigo_csr_organization	Conditional	The organization name which is included in the certificate **Organization (O)** field. Required if `sectigo_csr` is not provided.
sectigo_csr_organization_unit	Conditional	The organization unit which is included in the certificate **Organization Unit (OU)** field. Required if `sectigo_csr` is not provided.
sectigo_csr_email_address	Conditional	The email address which is included in the certificate **emailAddress** field. Required if `sectigo_csr` is not provided.
sectigo_csr_key_algo	Conditional	The private key algorithm to use to generate the private key. Default: RSA. Required if `sectigo_csr` is not provided.
sectigo_csr_key_size	Conditional	Size of the SSL/TLS key to generate. Possible values: • 2048-bit (default) • 3072-bit

		• 4096-bit Required if `sectigo_csr` is not provided.
sectigo_csr	Conditional	A certificate signing request PEM that users can optionally provide if they don't want to generate a new one.
sectigo_private_key	Conditional	A private key PEM that users can optionally provide if they want to generate a CSR by using it.

3.1.3 Certificate Issuance and Collection Parameters

Parameter	Type	Description
Common		
sectigo_max_timeout	Optional	The maximum time in seconds before a certificate download attempt will time out. Default is 600 seconds.
sectigo_loop_period	Optional	The time in seconds between each attempt to download the issued certificate. Default is 30 seconds.
sectigo_expiry_window	Optional	The period of days prior to expiration that a new certificate enrollment process will be initiated. Default is 7 days.
sectigo_auto_renew	Optional	A flag to determine whether certificates that fall within the expiry window should get automatically renewed. Default is **True**.
sectigo_read_renewed_cert	Optional	When this flag is set to **True** (default) and you attempt to read a certificate from Vault, if the certificate has the **Renewed** state, the newer certificate is returned instead of the requested one. If you set this flag to **False**, then even if the certificate had already been renewed, the requested certificate is returned. Specifying **False** may help avoid unwanted renewals.
SSL Certificates Only		
sectigo_ssl_cert_type	Mandatory	Type of SSL certificate (numeric). This is the ID of the SSL certificate type.

sectigo_ssl_cert_validity	Mandatory	Certificate validity period in days (numeric). The values available are dependent on the selected `sectigo_ssl_cert_type`.
sectigo_ssl_cert_external_requester	Optional	A comma-separated list of emails.
sectigo_ssl_cert_format_type	Optional	Format type for SSL certificate. The supported values are: • **x509**—for X509, Base64 encoded (default) • **x509CO**—for X509 Certificate only, Base64 encoded • **x509IO**—for X509 Intermidiates/Root only, Base64 encoded • **base64**—for PKCS#7 Base64 encoded • **bin**—for PKCS#7 Bin encoded • **x509IOR**—for X509 Intermediates/Root only Reverse, Base64 encoded
sectigo_ssl_cert_comments	Optional	Comments for certificate enrollment.
sectigo_ssl_cert_num_servers	Conditional	The number of server licenses (numeric).
sectigo_ssl_cert_server_type	Optional	The server type ID (numeric).
sectigo_ssl_cert_subject_alt_names	Optional	A comma-separated list of subject alternative names (SAN).
sectigo_ssl_cert_custom_fields[3]	Optional	Custom fields to be applied to the requested certificate.
Client Certificates Only		
sectigo_client_cert_type	Mandatory	Type of client certificate (numeric). This is the ID of the client certificate type.
sectigo_client_cert_validity	Mandatory	Certificate validity period in days (numeric). The values available are dependent on the selected `sectigo_client_cert_type`.
sectigo_client_cert_email	Mandatory	The user's email. Must be a valid email. Must be less than 256 characters.
sectigo_client_cert_first_name	Conditional	The user's first name.
sectigo_client_cert_middle_name	Conditional	The user's middle name.
sectigo_client_cert_last_name	Conditional	The user's last name.

		The combined length of the first, middle, and last name fields cannot exceed 64 characters.
sectigo_client_cert_custom_fields[3]	Optional	Custom fields to be applied to the requested certificate.

3.1.4 Other Parameters

Parameter	Type	Description
Common		
sectigo_cert_unique_id	Conditional	The unique certificate ID is used as the main identifier for certificates that are stored in Vault. For SSL certificates, the cert unique ID is `<ssl_id>_<customer_uri>`. For client certificates, the unique cert ID is `<order_number>_<customer_uri>`.
sectigo_reason	Mandatory	The reason why an action is being taken. • **revoke**—The reason why a certificate is being revoked. • **replace**—The reason why a certificate is to be replaced.
SSL Certificates Only		
sectigo_common_name	Conditional	Used in path `replace`. Single value for a domain that is included in the certificate **Common Name (CN)** field.
Client Certificates Only		
sectigo_client_cert_revoked_on_replace	Mandatory	Used in path `replace`. Flag to determine whether a replaced certificate should be revoked.

[3] The expected format for custom fields is the following: `[{"name":"custom_field_1","value":"value_1"},{"name":"custom_field_2","value":"value_2"}]`. If you are providing this input in a JSON String, make sure that the internal double quotes are escaped properly using \.

4 Using the Sectigo Vault PKI Plugin

4.1 Configuring the Plugin Directory

In order to use the Sectigo Vault PKI plugin, you must place the custom `sectigo-vault-pki_<version>` binary in your Vault plugins directory (for example, `/etc/vault/custom_plugins/`). Make sure that the custom plugin binary has the proper execute permissions enabled. On Linux, this can be done using the following command:

```
$ chmod +x /etc/vault/custom_plugins/sectigo-vault-pki_<version>
```

Additionally, your Vault server's configuration must have the `plugin_directory` field configured and pointing to the path of your `plugins` directory. For more information on the `plugin_directory` field, see https://www.vaultproject.io/docs/configuration/#plugin_directory.

4.2 Setting Up Environmental Variables

Whether you are planning on using the Vault CLI tool or cURL commands, you may want to export the following environment variables to facilitate your interactions with your Vault server:

```
$ export VAULT_ADDR='http://<vault_ip_address>:<vault_port_number>'
```

```
$ export
VAULT_API_ADDR='http://<vault_ip_address>:<vault_port_number>'
```

```
$ export VAULT_TOKEN='<token_goes_here>'
```

```
$ export VAULT_CLIENT_TIMEOUT='600'
```

Note: The address environment variables are shown for http. Make sure to use https instead of http on your production server.

4.3 Enabling the Plugin

Assuming you have a Vault server that is (1) running and unsealed, (2) configured to point to a plugin directory where the `sectigo-vault-pki` binary is located, and (3) accessible through the environment variables that you have exported, you may enable the Sectigo PKI plugin by running:

```
$ SHA256=$(shasum -a 256 <path_to_plugin_directory>/sectigo-vault-
pki_<version>| cut -d' ' -f1)
```

```
$ vault write sys/plugins/catalog/secret/sectigo-vault-pki_<version>
sha_256="${SHA256}" command=sectigo-vault-pki_<version>
```

```
$ vault secrets enable -path=sectigo-vault-pki -plugin-name=sectigo-
vault-pki_<version> sectigo-vault-pki_<version>
```

4.4 Interacting with the Plugin

Users may interact with Vault directly through the Vault CLI tool, or they may interact with the exposed API endpoints through a tool such a cURL. In this section we describe how to interact with the Sectigo PKI plugin using the Vault CLI tool. Please consult the `README.md` file for sample cURL commands.

In the downloaded package, you have a directory named `sample_data_jsons` which contains two subdirectories: `ssl_cert` and `client_cert`. Each of these subdirectories contains JSON files which correspond to the paths that are supported by the custom plugin. The commands below refer to the `ssl_cert` JSON files. For client certificates, simply use the `client_cert` JSON files instead.

Make sure to customize the variables in the applicable JSON files to match your SCM settings / certificate requirements.

Action	Command
Creating a Config Entry	`$ vault write sectigo-vault-pki/configs/<config_name> @ssl_cert_config.json`
Enrolling and Collecting a Certificate[4]	`$ vault write sectigo-vault-pki/enroll/<config_name> @ssl_cert.json`
Reading a Certificate	`$ vault read sectigo-vault-pki/certs/<config_name>/<sectigo_cert_unique_id> @ssl_cert.json`
Revoking a Certificate[5]	`$ vault write sectigo-vault-pki/revoke/<config_name>/<sectigo_cert_unique_id> @ssl_cert_revoke.json`
Replacing a Certificate	`$ vault write sectigo-vault-pki/replace/<config_name>/<sectigo_cert_unique_id> @ssl_cert_replace.json`
Fetching an Existing Certificate from SCM[6]	`$ vault write sectigo-vault-pki/fetch/<config_name>/<cert_id> @ssl_cert_fetch.json`
Deleting a Certificate from Vault	`$ vault delete sectigo-vault-pki/certs/<config_name>/<sectigo_cert_unique_id>`
Deleting a Config Entry from Vault	`$ vault delete sectigo-vault-pki/configs/<config_name>`

[4] This step can take a few minutes to complete and may be subject to different types of timeouts. For more information read the Dealing with Timeouts and Error Codes section.

[5] Revoking a certificate does not automatically delete it from Vault.

[6] For path `fetch`, `<cert_id>` is the SSL ID in the case of SSL certificates, or Order Number in the case of client certificates.

Listing All Certificates Under a Config Name	`$ vault list sectigo-vault-pki/certs/<config_name>`
Listing All Config Entries Stored in Vault	`$ vault list sectigo-vault-pki/configs/`
Reading a Config Entry	`$ vault read sectigo-vault-pki/configs/<config_name>`
Manually Renewing a Certificate	`$ vault write sectigo-vault-pki/renew/<config_name>/<sectigo_cert_unique_id> @ssl_cert_manual_renew.json`
Automatically Renewing a Certificate	The certificate validity check takes place each time a user reads a certificate that is stored in Vault. Automatic certificate renewal gets triggered when the following conditions are met: • The `sectigo_auto_renew` flag is set to **True**[7]. • The number of remaining days until certificate expiry falls within the user-specified `sectigo_expiry_window` field. For more information, see Certificate Renewal Flow Diagram.

4.5 Output

Using the Vault CLI tool, users may read entries from Vault using three different formats:

- Table
- JSON
- YAML

All cURL responses are returned using only the JSON format. For regular SSL certificates[8], the output for a typical certificate entry that is stored in Vault will have the following key/value pairs[9]:

[7] By default, the `sectigo_auto_renew` field is set to **True**.

[8] Client certificates have a very similar output to SSL certificates; client certificates do not have an `ssl_format` field. Moreover, instead of `ssl_id`, client certificates use `order_number`.

[9] This output is shown using the JSON format. The certificate-related data appears under the `data` JSON array. Fields outside of the `data` JSON array are internally set and used by Vault. The Sectigo Vault PKI plugin does not currently support Vault leases.

Sample JSON Output

```
{
  "request_id": "<request_id>",
  "lease_id": "",
  "renewable": false,
  "lease_duration": 0,
  "data": {
    "cert_unique_id": "<ssl_id/order_number>_<customer_uri>",
    "certificate": "<certificate>",
    "certificate_type": "<ssl_cert>/<client_cert>",
    "csr": "<csr>",
    "domain": "<domain>",
    "private_key": "<private_key>",
    "renew_id": "<renew_id>",
    "ssl_format": "<ssl_format>",
    "ssl_id": <ssl_id>
    "state": [
      {
        "status": "<state>",
        "time_stamp": "<time_stamp>"
      }
    ]
  },
  "wrap_info": null,
  "warnings": null,
  "auth": null
}
```

4.6 Logs

The Sectigo Vault PKI plugin prints operational logs directly on the Vault server logs. These logs are typically accessible through the `STDOUT` of the running Vault server. Users may increase/decrease the log level as per their requirements. For more information on Vault server logs and on changing the log level, see https://learn.hashicorp.com/vault/operations/troubleshooting-vault#vault-logs.

4.7 How to Use Existing CSRs and/or Private Keys

When enrolling a certificate, users have the option to provide an existing private key and/or an existing CSR. If either of these two values are provided, the Sectigo Vault PKI plugin will use them instead of generating new ones.

There are two main techniques to enroll a certificate using input from existing CSRs and/or private key PEM files[10]:

- Passing CSRs/Private Keys as PEM Files
- Passing CSRs/Private Keys as Strings

[10] These techniques also apply to paths other than `enroll` that can handle taking a CSR or a private key as input (e.g. path `replace`).

4.7.1 Passing CSRs/Private Keys as PEM Files

1. Customize the entries in `ssl_cert.json` to match your requirements. Do not include the `sectigo_csr` and `sectigo_private_key` variables in the JSON input[11].
2. Execute the following (one-line) command. Make sure to provide a valid `config_name`, and point to the `csr.pem` and `private_key.pem` files that exist on your machine.

```
$ vault write sectigo-vault-pki/enroll/<config_name>
sectigo_csr=@csr.pem sectigo_private_key=@private_key.pem
@ssl_cert.json
```

4.7.2 Passing CSRs/Private Keys as Strings

1. Customize the entries in `ssl_cert.json` to match your requirements.
   - Provide the input for `sectigo_csr` and/or `sectigo_private_key` variables in the applicable JSON file.
   - Ensure that the PEM strings are escaped properly (use `\n` instead of separate lines for each individual line from the PEM String; see the sample JSON CSR).
2. Execute the following (one-line) command. Make sure that the `config_name` that you created in the previous step is passed in the enroll path.

```
$ vault write sectigo-vault-pki/enroll/<config_name> @ssl_cert.json
```

Sample JSON CSR

```
  "sectigo_csr": "-----BEGIN CERTIFICATE REQUEST-----
\nMIIC6jCCAdICAQAwgYYxCzAJBgNVBAYTAkNBMQswCQYDVQQIEwJPTjEPMA0GA1UE\nBxMGS2FuYXR
hMRAwDgYDVQQKEwdTZWN0aWdvMQ8wDQYDVQQLEwZEZXZPcHMxEjAQ\nBgNVBAMTCWNjbXFhLmNvbTEi
MCAGCSqGSIb3DQEJARYTZGVtb3VzZXIxQGNjbXFh\nLmNvbTCCASIwDQYJKoZIhvcNAQEBBQADggEPA
DCCAQoCggEBAJv/fimH2Iws+BSW\nznqYkXN2QYP/Qt8RA5bfehcQvp8vBItt+bFP7J30aUMLtEgz+G
hcrVTd5yhx9Unk\nVEPr5u/t6Zbr+KgJm2BOLUNXVCg/xK7KxhWMdXGuGxluQtMIE48CY9Xj3LNPVmV
U\nb1CFXNq1c0BJjJJdqrr3n1lKw4g3Je2xPNKbBOjJte33N6fRZIyuBrM+SmLMF6sY\nj2THadd8L2
KwYtVHSV3lYgg7dmw/EL408EWt5raH0WJix0srnEf/4R67zljoAdoD\n+wQiUs0oXzuyOJ7i7Uv2gsf
VyqYTYsscpeAvGBkdJoQcWTbKzAdUYjgFZ4nhHY74\n0ANHcesCAwEAAaAeMBwGCSqGSIb3DQEJDjEP
MA0wCwYDVR0RBAQwAoIAMA0GCSqG\nSIb3DQEBCwUAA4IBAQBwEBJllP1isYP3pgrft+VErV+YY8wXB
bY5FD//Jasq2ziz\nf717wdCQoVyznoVz8wRvX16BpllXAdxmysFb+1F2HwoZywQ6VCPmmgioaJv9HL
kS\nFY7ZYeRJhE+aEHlI2JaULnynHrZarMFtBMoPlatmhDLHUFgBWy9RHt38TxxZg893\nszDEqWYYF
5q1GXCGWL6TRbbaZHvCLWMzNqpNGWaInw2CSbdi6/Wd5vIT7BKWc/06\nggVGOHXRNF2O5a4ImUMz96
uh6N9c6MH+BBivo77ZjCK0qbKTIKtnnimxlQnn0vht\nEqWTyOJwsAkzEgTqpE5h3I6boh4KVJjhK8c
QVsQL\n-----END CERTIFICATE REQUEST-----\n"
```

4.8 Dealing with Timeouts and Error Codes

The certificate issuance process on SCM can often take a few minutes to complete. It is therefore important to properly handle possible timeout situations to prevent unwanted loss of newly enrolled, replaced, or renewed certificates.

[11] The Vault CLI tool does not allow you to override the value of a variable that's specified in a JSON input file with a non-JSON value. If you include the variables for `sectigo_csr` and `sectigo_private_key` in your JSON file and attempt to override the input using non-JSON input, you will get a *failed to parse* error message from Vault.

There are three types of timeouts that you may encounter when attempting to collect a certificate that is in the **Applied** state on SCM:

- Timeouts that are related to the `sectigo_max_timeout` parameter that's configured in your applicable JSON file.
- Timeouts that are set on the configured listener on your Vault server. See full list of listener parameters here.
- Timeouts that are related to the client tool that you are using (such as, Vault CLI tool or cURL). For the Vault CLI tool, the `VAULT_CLIENT_TIMEOUT` environment variable can be used. Vault 1.2.3 had a bug that prevented the Vault server from picking up that environment variable, that bug was fixed in Vault 1.3.0 (the changelog details for that release can be accessed here).

To avoid issues due to timeouts, make sure to either increase your Vault server and client timeouts, or to reduce the `sectigo_max_timeout` in your applicable JSON file such that it is smaller than your existing Vault server and client timeouts. In case your download timed out due to it requiring more time than specified in `sectigo_max_timeout`, and your certificate result contained `TimedoutStateSaved`, re-reading the certificate will make the Sectigo Vault PKI plugin re-attempt to collect it from SCM. If a Vault server or client timeout occurs and you are not sure if your certificate got stored in Vault, use the `vault list sectigo-vault-pki/certs/<config_name>` command to see all certificates that are stored under your given config name.

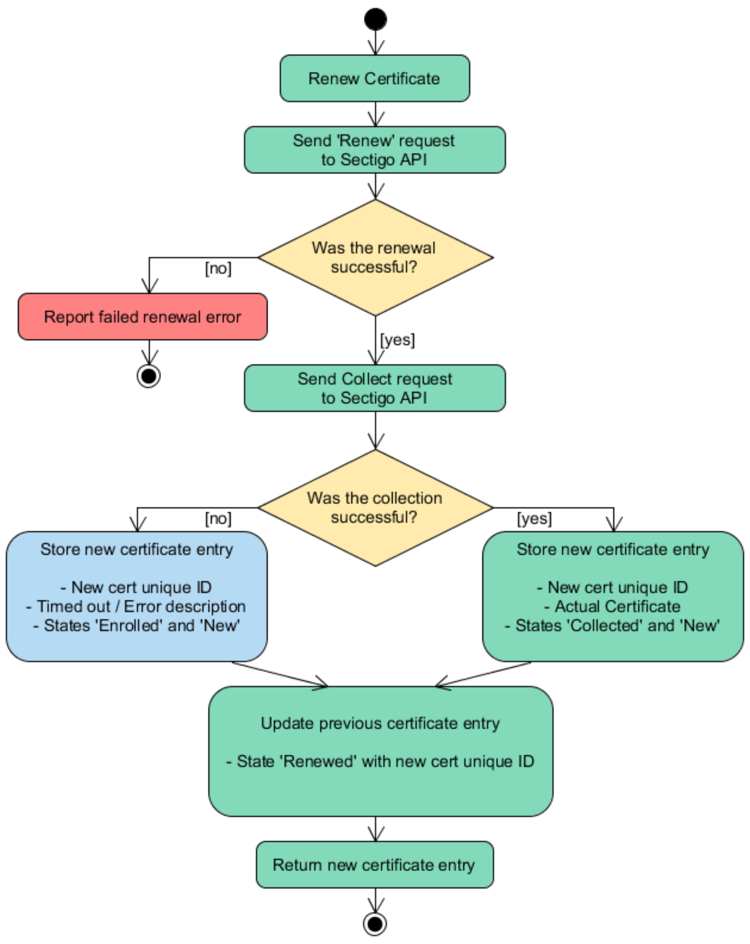
In some cases, it is possible to successfully enroll, replace, or renew a certificate on SCM and still face an issue when attempting to collect it (for example, if you are required to provide further manual approval for the certificate). In such cases, the certificate result will be set to `ErrorCode` along with the description that's given by SCM. When this happens, read the given `ErrorCode` and try to fix the shown issue. Once you have fixed the applicable issue, re-reading the certificate will make the Sectigo Vault PKI plugin re-attempt to collect it from SCM.

5 Additional Notes

5.1 Understanding Path Certs and Path Fetch

The `certs` path implemented in the Sectigo Vault PKI plugin allows you to read, delete, or list certificates that are already stored in Vault under the given config name. The `fetch` path gives you the ability to download certificates, that are not already stored in Vault, from SCM and to store them on your Vault server. If a user attempts to `fetch` a certificate that already exists in Vault, the behavior of the command will be the same as reading a certificate through path `certs`.

5.2 Certificate Renewal Flow Diagram


Renew Certificate
Send 'Renew' request to Sectigo API
Was the renewal successful?
[no]
Report failed renewal error
[yes]
Send Collect request to Sectigo API
Was the collection successful?
[no]
[yes]
Store new certificate entry - New cert unique ID - Timed out / Error description - States 'Enrolled' and 'New'
Store new certificate entry - New cert unique ID - Actual Certificate - States 'Collected' and 'New'
Update previous certificate entry - State 'Renewed' with new cert unique ID
Return new certificate entry