PKI Secrets Engine (API)

This is the API documentation for the Vault PKI secrets engine. For general information about the usage and operation of the PKI secrets engine, please see the PKI documentation.

This documentation assumes the PKI secrets engine is enabled at the `/pki` path in Vault. Since it is possible to enable secrets engines at any location, please update your API calls accordingly.

Table of Contents

Read CA Certificate

This endpoint retrieves the CA certificate *in raw DER-encoded form*. This is a bare endpoint that does not return a standard Vault data structure and cannot be read by the Vault CLI; use `/pki/cert` for that. If `/pem` is added to the endpoint, the CA certificate is returned in PEM format.

This is an unauthenticated endpoint.

Method	Path	Produces
`GET`	`/pki/ca(/pem)`	`200 application/binary`

Sample Request

```
$ curl \
    http://127.0.0.1:8200/v1/pki/ca/pem
```

Sample Response

```
<binary DER-encoded certificate>
```

Read CA Certificate Chain

This endpoint retrieves the CA certificate chain, including the CA *in PEM format*. This is a bare endpoint that does not return a standard Vault data structure and cannot be read by the Vault CLI; use `/pki/cert` for that.

This is an unauthenticated endpoint.

Method	Path	Produces
`GET`	`/pki/ca_chain`	`200 application/binary`

Sample Request

```
$ curl \
    http://127.0.0.1:8200/v1/pki/ca_chain
```

Sample Response

```
<PEM-encoded certificate chain>
```

Read Certificate

This endpoint retrieves one of a selection of certificates. This endpoint returns the certificate in PEM formatting in the `certificate` key of the JSON object, which is a standard Vault response that is readable by the Vault CLI.

This is an unauthenticated endpoint.

Method	Path	Produces
`GET`	`/pki/cert/:serial`	`200 application/json`

Parameters

- `serial` `(string: <required>)` – Specifies the serial of the key to read. This is part of the request URL. Valid values for `serial` are:
  - `<serial>` for the certificate with the given serial number
  - `ca` for the CA certificate
  - `crl` for the current CRL
  - `ca_chain` for the CA trust chain or a serial number in either hyphen-separated or colon-separated octal format

Sample Request

```
$ curl \
    http://127.0.0.1:8200/v1/pki/cert/crl
```

Sample Response

```
{
  "data": {
    "certificate": "-----BEGIN CERTIFICATE-----\nMIIGmDCCBYCgAwIBAgIHBzEB3fTzhTANBgkqhkiG9w0BAQsFADCBjDELMAkGA1UE\n..."
  }
}
```

List Certificates

This endpoint returns a list of the current certificates by serial number only.

Method	Path	Produces
`LIST`	`/pki/certs`	`200 application/json`

Sample Request

```
$ curl \
    --header "X-Vault-Token: ..." \
    --request LIST \
    http://127.0.0.1:8200/v1/pki/certs
```

Sample Response

```
{
  "lease_id":"",
  "renewable":false,
  "lease_duration":0,
  "data":{
    "keys":[
      "17:67:16:b0:b9:45:58:c0:3a:29:e3:cb:d6:98:33:7a:a6:3b:66:c1",
      "26:0f:76:93:73:cb:3f:a0:7a:ff:97:85:42:48:3a:aa:e5:96:03:21"
    ]
  },
  "wrap_info":null,
  "warnings":null,
  "auth":null
}
```

Submit CA Information

This endpoint allows submitting the CA information for the backend via a PEM file containing the CA certificate and its private key, concatenated.

May optionally append additional CA certificates. Useful when creating an intermediate CA to ensure a full chain is returned when signing or generating certificates.

Not needed if you are generating a self-signed root certificate, and not used if you have a signed intermediate CA certificate with a generated key (use the `/pki/intermediate/set-signed` endpoint for that). *If you have already set a certificate and key, they will be overridden.*

Method	Path	Produces
`POST`	`/pki/config/ca`	`204 (empty body)`

Parameters

- `pem_bundle` `(string: <required>)` – Specifies the key and certificate concatenated in PEM format.

Sample Request

```
$ curl \
    --header "X-Vault-Token: ..." \
    --request POST \
    --data "@payload.json" \
    http://127.0.0.1:8200/v1/pki/config/ca
```

Note that if you provide the data through the HTTP API it must be JSON-formatted, with newlines replaced with `\n` , like so:

```
{
  "pem_bundle": "-----BEGIN RSA PRIVATE KEY-----\n...\n-----END CERTIFICATE-----"
}
```

Read CRL Configuration

This endpoint allows getting the duration for which the generated CRL should be marked valid.

Method	Path	Produces
GET	/pki/config/crl	200 application/json

Sample Request

```
$ curl \
    --header "X-Vault-Token: ..." \
    http://127.0.0.1:8200/v1/pki/config/crl
```

Sample Response

```
{
  "lease_id": "",
  "renewable": false,
  "lease_duration": 0,
  "data": {
      "disable": false,
      "expiry": "72h"
    },
  "auth": null
}
```

Set CRL Configuration

This endpoint allows setting the duration for which the generated CRL should be marked valid. If the CRL is disabled, it will return a signed but zero-length CRL for any request. If enabled, it will re-build the CRL.

Note: Disabling the CRL does not affect whether revoked certificates are stored internally. Certificates that have been revoked when a role's certificate storage is enabled will continue to be marked and stored as revoked until `tidy` has been run with the desired safety buffer. Re-enabling CRL generation will then result in all such certificates becoming a part of the CRL.

Method	Path	Produces
`POST`	`/pki/config/crl`	`204 (empty body)`

Parameters

- `expiry` `(string: "72h")` – Specifies the time until expiration.
- `disable` `(bool: false)` – Disables or enables CRL building.

Sample Payload

```
{
  "expiry": "48h"
}
```

Sample Request

```
$ curl \
    --header "X-Vault-Token: ..." \
    --request POST \
    --data @payload.json \
    http://127.0.0.1:8200/v1/pki/config/crl
```

Read URLs

This endpoint fetches the URLs to be encoded in generated certificates.

Method	Path	Produces
`GET`	`/pki/config/urls`	`200 application/json`

Sample Request

```
$ curl \
    --header "X-Vault-Token: ..." \
    http://127.0.0.1:8200/v1/pki/config/urls
```

Sample Response

```
{
  "lease_id": "",
  "renewable": false,
  "lease_duration": 0,
  "data": {
    "issuing_certificates": ["<url1>", "<url2>"],
    "crl_distribution_points": ["<url1>", "<url2>"],
    "ocsp_servers": ["<url1>", "<url2>"]
  },
  "auth": null
}
```

Set URLs

This endpoint allows setting the issuing certificate endpoints, CRL distribution points, and OCSP server endpoints that will be encoded into issued certificates. You can update any of the values at any time without affecting the other existing values. To remove the values, simply use a blank string as the parameter.

Method	Path	Produces
`POST`	`/pki/config/urls`	`204 (empty body)`

Parameters

- `issuing_certificates` `(array<string>: nil)` – Specifies the URL values for the Issuing Certificate field. This can be an array or a comma-separated string list.

- `crl_distribution_points` `(array<string>: nil)` – Specifies the URL values for the CRL Distribution Points field. This can be an array or a comma-separated string list.

- `ocsp_servers` `(array<string>: nil)` – Specifies the URL values for the OCSP Servers field. This can be an array or a comma-separated string list.

Sample Payload

```
{
  "ocsp_servers": ["https://..."]
}
```

Sample Request

```
$ curl \
    --header "X-Vault-Token: ..." \
    --request POST \
    --data @payload.json \
    http://127.0.0.1:8200/v1/pki/config/urls
```

Read CRL

This endpoint retrieves the current CRL **in raw DER-encoded form**. This endpoint is suitable for usage in the CRL Distribution Points extension in a CA certificate. This is a bare endpoint that does not return a standard Vault data structure and cannot be parsed by the Vault CLI; use `/pki/cert/crl` in that case. If `/pem` is added to the endpoint, the CRL is returned in PEM format.

This is an unauthenticated endpoint.

Method	Path	Produces
`GET`	`/pki/crl(/pem)`	`200 application/binary`

Sample Request

```
$ curl \
    http://127.0.0.1:8200/v1/pki/crl/pem
```

Sample Response

```
<binary DER-encoded CRL>
```

Rotate CRLs

This endpoint forces a rotation of the CRL. This can be used by administrators to cut the size of the CRL if it contains a number of certificates that have now expired, but has not been rotated due to no further certificates being revoked.

Method	Path	Produces
`GET`	`/pki/crl/rotate`	`200 application/json`

Sample Request

```
$ curl \
    --header "X-Vault-Token: ..." \
    http://127.0.0.1:8200/v1/pki/crl/rotate
```

Sample Response

```
{
  "data": {
    "success": true
  }
}
```

Generate Intermediate

This endpoint generates a new private key and a CSR for signing. If using Vault as a root, and for many other CAs, the various parameters on the final certificate are set at signing time and may or may not honor the parameters set here. *This will overwrite any previously existing CA private key.*

This is mostly meant as a helper function, and not all possible parameters that can be set in a CSR are supported.

Method	Path	Produces
`POST`	`/pki/intermediate/generate/:type`	`200 application/json`

Parameters

- `type` `(string: <required>)` – Specifies the type of the intermediate to create. If `exported`, the private key will be returned in the response; if `internal` the private key will not be returned and *cannot be retrieved later*. This is part of the request URL.
- `common_name` `(string: <required>)` – Specifies the requested CN for the certificate.
- `alt_names` `(string: "")` – Specifies the requested Subject Alternative Names, in a comma-delimited list. These can be host names or email addresses; they will be parsed into their respective fields.
- `ip_sans` `(string: "")` – Specifies the requested IP Subject Alternative Names, in a comma-delimited list.
- `uri_sans` `(string: "")` – Specifies the requested URI Subject Alternative Names, in a comma-delimited list.
- `other_sans` `(string: "")` – Specifies custom OID/UTF8-string SANs. These must match values specified on the role in `allowed_other_sans` (globbing allowed). The format is the same as OpenSSL: `<oid>;<type>:<value>` where the only current valid type is `UTF8`. This can be a comma-delimited list or a JSON string slice.
- `format` `(string: "")` – Specifies the format for returned data. This can be `pem`, `der`, or `pem_bundle`; defaults to `pem`. If `der`, the output is base64 encoded. If `pem_bundle`, the `csr` field will contain the private key (if exported) and CSR, concatenated.
- `private_key_format` `(string: "")` – Specifies the format for marshaling the private key. Defaults to `der` which will return either base64-encoded DER or PEM-encoded DER, depending on the value of `format`. The other option is `pkcs8` which will

return the key marshalled as PEM-encoded PKCS8.

- `key_type` `(string: "rsa")` – Specifies the desired key type; must be `rsa` or `ec` .
- `key_bits` `(int: 2048)` – Specifies the number of bits to use. This must be changed to a valid value if the `key_type` is `ec` .
- `exclude_cn_from_sans` `(bool: false)` – If true, the given `common_name` will not be included in DNS or Email Subject Alternate Names (as appropriate). Useful if the CN is not a hostname or email address, but is instead some human-readable identifier.
- `ou` `(string: "")` – Specifies the OU (OrganizationalUnit) values in the subject field of the resulting CSR. This is a comma-separated string or JSON array.
- `organization` `(string: "")` – Specifies the O (Organization) values in the subject field of the resulting CSR. This is a comma-separated string or JSON array.
- `country` `(string: "")` – Specifies the C (Country) values in the subject field of the resulting CSR. This is a comma-separated string or JSON array.
- `locality` `(string: "")` – Specifies the L (Locality) values in the subject field of the resulting CSR. This is a comma-separated string or JSON array.
- `province` `(string: "")` – Specifies the ST (Province) values in the subject field of the resulting CSR. This is a comma-separated string or JSON array.
- `street_address` `(string: "")` – Specifies the Street Address values in the subject field of the resulting CSR. This is a comma-separated string or JSON array.
- `postal_code` `(string: "")` – Specifies the Postal Code values in the subject field of the resulting CSR. This is a comma-separated string or JSON array.

Sample Payload

```
{
  "common_name": "www.example.com"
}
```

Sample Request

```
$ curl \
    --header "X-Vault-Token: ..." \
    --request POST \
    --data @payload.json \
    http://127.0.0.1:8200/v1/pki/intermediate/generate/internal
```

```
{
  "lease_id": "",
  "renewable": false,
  "lease_duration": 21600,
  "data": {
    "csr": "-----BEGIN CERTIFICATE REQUEST-----\nMIIDzDCCAragAwIBAgIUOd0ukLcjH43TfTHFG9qE0FtlMVgwCwYJKoZIhvcNAQEL\n...\numkqe
    "private_key": "-----BEGIN RSA PRIVATE KEY-----\\nMIIEpAIBAAKCAQEAwsANtGz9gS3o5SwTSlOG1l-----END RSA PRIVATE KEY-----",
    "private_key_type": "rsa"
  },
  "warnings": null,
  "auth": null
}
```

Set Signed Intermediate

This endpoint allows submitting the signed CA certificate corresponding to a private key generated via `/pki/intermediate/generate` . The certificate should be submitted in PEM format; see the documentation for `/pki/config/ca` for some hints on submitting.

Method	Path	Produces
POST	/pki/intermediate/set-signed	204 (empty body)

Parameters

- `certificate` `(string: <required>)` – Specifies the certificate in PEM format. May optionally append additional CA certificates to populate the whole chain, which will then enable returning the full chain from issue and sign operations.

Sample Payload

```
{
  "certificate": "-----BEGIN CERTIFICATE..."
}
```

Sample Request

```
$ curl \
    --header "X-Vault-Token: ..." \
    --request POST \
    --data @payload.json \
    http://127.0.0.1:8200/v1/pki/intermediate/set-signed
```

Generate Certificate

This endpoint generates a new set of credentials (private key and certificate) based on the role named in the endpoint. The issuing CA certificate is returned as well, so that only the root CA need be in a client's trust store.

The private key is *not* stored. If you do not save the private key, you will need to request a new certificate.

Method	Path	Produces
`POST`	`/pki/issue/:name`	`200 application/json`

Parameters

- `name` `(string: <required>)` – Specifies the name of the role to create the certificate against. This is part of the request URL.
- `common_name` `(string: <required>)` – Specifies the requested CN for the certificate. If the CN is allowed by role policy, it will be issued.
- `alt_names` `(string: "")` – Specifies requested Subject Alternative Names, in a comma-delimited list. These can be host names or email addresses; they will be parsed into their respective fields. If any requested names do not match role policy, the entire request will be denied.
- `ip_sans` `(string: "")` – Specifies requested IP Subject Alternative Names, in a comma-delimited list. Only valid if the role allows IP SANs (which is the default).
- `uri_sans` `(string: "")` – Specifies the requested URI Subject Alternative Names, in a comma-delimited list.
- `other_sans` `(string: "")` – Specifies custom OID/UTF8-string SANs. These must match values specified on the role in `allowed_other_sans` (globbing allowed). The format is the same as OpenSSL: `<oid>;<type>:<value>` where the only current valid type is `UTF8` . This can be a comma-delimited list or a JSON string slice.
- `ttl` `(string: "")` – Specifies requested Time To Live. Cannot be greater than the role's `max_ttl` value. If not provided, the role's `ttl` value will be used. Note that the role values default to system values if not explicitly set.
- `format` `(string: "")` – Specifies the format for returned data. Can be `pem` , `der` , or `pem_bundle` ; defaults to `pem` . If `der` , the output is base64 encoded. If `pem_bundle` , the `certificate` field will contain the private key and certificate, concatenated; if the issuing CA is not a Vault-derived self-signed root, this will be included as well.
- `private_key_format` `(string: "")` – Specifies the format for marshaling the private key. Defaults to `der` which will return either base64-encoded DER or PEM-encoded DER, depending on the value of `format` . The other option is `pkcs8` which will return the key marshalled as PEM-encoded PKCS8.
- `exclude_cn_from_sans` `(bool: false)` – If true, the given `common_name` will not be included in DNS or Email Subject Alternate Names (as appropriate). Useful if the CN is not a hostname or email address, but is instead some human-readable identifier.

Sample Payload

```
{
  "common_name": "www.example.com"
}
```

Sample Request

```
$ curl \
    --header "X-Vault-Token: ..." \
    --request POST \
    --data @payload.json \
    http://127.0.0.1:8200/v1/pki/issue/my-role
```

Sample Response

```
{
  "lease_id": "pki/issue/test/7ad6cfa5-f04f-c62a-d477-f33210475d05",
  "renewable": false,
  "lease_duration": 21600,
  "data": {
    "certificate": "-----BEGIN CERTIFICATE-----\nMIIDzDCCAragAwIBAgIUOd0ukLcjH43TfTHFG9qE0FtlMVgwCwYJKoZIhvcNAQEL\n...\numkqeY
    "issuing_ca": "-----BEGIN CERTIFICATE-----\nMIIDUTCCAjmgAwIBAgIJAKM+z4MSfw2mMA0GCSqGSIb3DQEBCwUAMBsxGTAXBgNV\n...\nG/7g4ko
    "ca_chain": ["-----BEGIN CERTIFICATE-----\nMIIDUTCCAjmgAwIBAgIJAKM+z4MSfw2mMA0GCSqGSIb3DQEBCwUAMBsxGTAXBgNV\n...\nG/7g4koc
    "private_key": "-----BEGIN RSA PRIVATE KEY-----\nMIIEowIBAAKCAQEAnVHfwoKsUG1GDVyWB1AFroaKl2ImMBO8EnvGLRrmobIkQvh+\n...\nQN
    "private_key_type": "rsa",
    "serial_number": "39:dd:2e:90:b7:23:1f:8d:d3:7d:31:c5:1b:da:84:d0:5b:65:31:58"
    },
  "warnings": "",
  "auth": null
}
```

Revoke Certificate

This endpoint revokes a certificate using its serial number. This is an alternative option to the standard method of revoking using Vault lease IDs. A successful revocation will rotate the CRL.

Method	Path	Produces
`POST`	`/pki/revoke`	`200 application/json`

Parameters

- `serial_number` `(string: <required>)` – Specifies the serial number of the certificate to revoke, in hyphen-separated or colon-separated octal.

Sample Payload

```
{
  "serial_number": "39:dd:2e..."
}
```

Sample Request

```
$ curl \
    --header "X-Vault-Token: ..." \
    --request POST \
    --data @payload.json \
    http://127.0.0.1:8200/v1/pki/revoke
```

Sample Response

```
{
  "data": {
    "revocation_time": 1433269787
  }
}
```

Create/Update Role

This endpoint creates or updates the role definition. Note that the `allowed_domains` , `allow_subdomains` , `allow_glob_domains` , and `allow_any_name` attributes are additive; between them nearly and across multiple roles nearly any issuing policy can be accommodated. `server_flag` , `client_flag` , and `code_signing_flag` are additive as well. If a client requests a certificate that is not allowed by the CN policy in the role, the request is denied.

Method	Path	Produces
`POST`	`/pki/roles/:name`	`204 (empty body)`

Parameters

- `name` `(string: <required>)` – Specifies the name of the role to create. This is part of the request URL.
- `ttl` `(string: "")` – Specifies the Time To Live value provided as a string duration with time suffix. Hour is the largest suffix. If not set, uses the system default value or the value of `max_ttl` , whichever is shorter.
- `max_ttl` `(string: "")` – Specifies the maximum Time To Live provided as a string duration with time suffix. Hour is the largest suffix. If not set, defaults to the system maximum lease TTL.
- `allow_localhost` `(bool: true)` – Specifies if clients can request certificates for `localhost` as one of the requested common names. This is useful for testing and to allow clients on a single host to talk securely.
- `allowed_domains` `(list: [])` – Specifies the domains of the role. This is used with the `allow_bare_domains` and `allow_subdomains` options.
- `allow_bare_domains` `(bool: false)` – Specifies if clients can request certificates matching the value of the actual domains themselves; e.g. if a configured domain set with `allowed_domains` is `example.com` , this allows clients to actually request a certificate containing the name `example.com` as one of the DNS values on the final certificate. In some scenarios, this can be considered a security risk.

- `allow_subdomains` `(bool: false)` – Specifies if clients can request certificates with CNs that are subdomains of the CNs allowed by the other role options. *This includes wildcard subdomains.* For example, an `allowed_domains` value of `example.com` with this option set to true will allow `foo.example.com` and `bar.example.com` as well as `*.example.com`. This is redundant when using the `allow_any_name` option.

- `allow_glob_domains` `(bool: false)` - Allows names specified in `allowed_domains` to contain glob patterns (e.g. `ftp*.example.com`). Clients will be allowed to request certificates with names matching the glob patterns.

- `allow_any_name` `(bool: false)` – Specifies if clients can request any CN. Useful in some circumstances, but make sure you understand whether it is appropriate for your installation before enabling it.

- `enforce_hostnames` `(bool: true)` – Specifies if only valid host names are allowed for CNs, DNS SANs, and the host part of email addresses.

- `allow_ip_sans` `(bool: true)` – Specifies if clients can request IP Subject Alternative Names. No authorization checking is performed except to verify that the given values are valid IP addresses.

- `allowed_uri_sans` `(string: "")` - Defines allowed URI Subject Alternative Names. No authorization checking is performed except to verify that the given values are valid URIs. This can be a comma-delimited list or a JSON string slice. Values can contain glob patterns (e.g. `spiffe://hostname/*`).

- `allowed_other_sans` `(string: "")` – Defines allowed custom OID/UTF8-string SANs. This field supports globbing. The format is the same as OpenSSL: `<oid>;<type>:<value>` where the only current valid type is `UTF8` (or `UTF-8`). This can be a comma-delimited list or a JSON string slice. All values, including globbing values, must use the correct syntax, with the exception being a single `*` which allows any OID and any value (but type must still be UTF8).

- `server_flag` `(bool: true)` – Specifies if certificates are flagged for server use.

- `client_flag` `(bool: true)` – Specifies if certificates are flagged for client use.

- `code_signing_flag` `(bool: false)` – Specifies if certificates are flagged for code signing use.

- `email_protection_flag` `(bool: false)` – Specifies if certificates are flagged for email protection use.

- `key_type` `(string: "rsa")` – Specifies the type of key to generate for generated private keys and the type of key expected for submitted CSRs. Currently, `rsa` and `ec` are supported, or when signing CSRs `any` can be specified to allow keys of either type and with any bit size (subject to > 1024 bits for RSA keys).

- `key_bits` `(int: 2048)` – Specifies the number of bits to use for the generated keys. This will need to be changed for `ec` keys. See https://golang.org/pkg/crypto/elliptic/#Curve for an overview of allowed bit lengths for `ec`.

- `key_usage` `(list: ["DigitalSignature", "KeyAgreement", "KeyEncipherment"])` – Specifies the allowed key usage constraint on issued certificates. Valid values can be found at https://golang.org/pkg/crypto/x509/#KeyUsage - simply drop the `KeyUsage` part of the value. Values are not case-sensitive. To specify no key usage constraints, set this to an empty list.

- `ext_key_usage` `(list: [])` – Specifies the allowed extended key usage constraint on issued certificates. Valid values can be found at https://golang.org/pkg/crypto/x509/#ExtKeyUsage - simply drop the `ExtKeyUsage` part of the value. Values are not case-sensitive. To specify no key usage constraints, set this to an empty list.

- `use_csr_common_name` `(bool: true)` – When used with the CSR signing endpoint, the common name in the CSR will be used instead of taken from the JSON data. This does `not` include any requested SANs in the CSR; use `use_csr_sans` for that.

- `use_csr_sans` `(bool: true)` – When used with the CSR signing endpoint, the subject alternate names in the CSR will be used instead of taken from the JSON data. This does `not` include the common name in the CSR; use `use_csr_common_name` for that.

- `ou` `(string: "")` – Specifies the OU (OrganizationalUnit) values in the subject field of issued certificates. This is a comma-separated string or JSON array.

- `organization` `(string: "")` – Specifies the O (Organization) values in the subject field of issued certificates. This is a comma-separated string or JSON array.
- `country` `(string: "")` – Specifies the C (Country) values in the subject field of issued certificates. This is a comma-separated string or JSON array.
- `locality` `(string: "")` – Specifies the L (Locality) values in the subject field of issued certificates. This is a comma-separated string or JSON array.
- `province` `(string: "")` – Specifies the ST (Province) values in the subject field of issued certificates. This is a comma-separated string or JSON array.
- `street_address` `(string: "")` – Specifies the Street Address values in the subject field of issued certificates. This is a comma-separated string or JSON array.
- `postal_code` `(string: "")` – Specifies the Postal Code values in the subject field of issued certificates. This is a comma-separated string or JSON array.
- `generate_lease` `(bool: false)` – Specifies if certificates issued/signed against this role will have Vault leases attached to them. Certificates can be added to the CRL by `vault revoke <lease_id>` when certificates are associated with leases. It can also be done using the `pki/revoke` endpoint. However, when lease generation is disabled, invoking `pki/revoke` would be the only way to add the certificates to the CRL. When large number of certificates are generated with long lifetimes, it is recommended that lease generation be disabled, as large amount of leases adversely affect the startup time of Vault.
- `no_store` `(bool: false)` – If set, certificates issued/signed against this role will not be stored in the storage backend. This can improve performance when issuing large numbers of certificates. However, certificates issued in this way cannot be enumerated or revoked, so this option is recommended only for certificates that are non-sensitive, or extremely short-lived. This option implies a value of `false` for `generate_lease` .
- `require_cn` `(bool: true)` - If set to false, makes the `common_name` field optional while generating a certificate.
- `policy_identifiers` `(list: [])` – A comma-separated string or list of policy OIDs.
- `basic_constraints_valid_for_non_ca` `(bool: false)` - Mark Basic Constraints valid when issuing non-CA certificates.
- `not_before_duration` `(duration: "30s")` – Specifies the duration by which to backdate the NotBefore property.

Sample Payload

```
{
  "allowed_domains": ["example.com"],
  "allow_subdomains": true
}
```

Sample Request

```
$ curl \
    --header "X-Vault-Token: ..." \
    --request POST \
    --data @payload.json \
    http://127.0.0.1:8200/v1/pki/roles/my-role
```

Read Role

This endpoint queries the role definition.

Method	Path	Produces
`GET`	`/pki/roles/:name`	`200 application/json`

Parameters

- `name` `(string: <required>)` – Specifies the name of the role to read. This is part of the request URL.

Sample Request

```
$ curl \
    --header "X-Vault-Token: ..." \
    http://127.0.0.1:8200/v1/pki/roles/my-role
```

Sample Response

```
{
  "data": {
    "allow_any_name": false,
    "allow_ip_sans": true,
    "allow_localhost": true,
    "allow_subdomains": false,
    "allowed_domains": ["example.com", "foobar.com"],
    "allow_uri_sans": ["example.com","spiffe://*"],
    "client_flag": true,
    "code_signing_flag": false,
    "key_bits": 2048,
    "key_type": "rsa",
    "ttl": "6h",
    "max_ttl": "12h",
    "server_flag": true
  }
}
```

List Roles

This endpoint returns a list of available roles. Only the role names are returned, not any values.

Method	Path	Produces
`LIST`	`/pki/roles`	`200 application/json`

Sample Request

```
$ curl \
    --header "X-Vault-Token: ..." \
    --request LIST \
    http://127.0.0.1:8200/v1/pki/roles
```

Sample Response

```
{
  "auth": null,
  "data": {
    "keys": ["dev", "prod"]
  },
  "lease_duration": 2764800,
  "lease_id": "",
  "renewable": false
}
```

Delete Role

This endpoint deletes the role definition. Deleting a role **does not** revoke certificates previously issued under this role.

Method	Path	Produces
`DELETE`	`/pki/roles/:name`	`204 (empty body)`

Parameters

- `name (string: <required>)` – Specifies the name of the role to delete. This is part of the request URL.

Sample Request

```
$ curl \
    --header "X-Vault-Token: ..." \
    --request DELETE \
    http://127.0.0.1:8200/v1/pki/roles/my-role
```

Generate Root

This endpoint generates a new self-signed CA certificate and private key. If the path ends with `exported`, the private key will be returned in the response; if it is `internal` the private key will not be returned and *cannot be retrieved later*. Distribution points use the values set via `config/urls`.

As with other issued certificates, Vault will automatically revoke the generated root at the end of its lease period; the CA certificate will sign its own CRL.

As of Vault 0.8.1, if a CA cert/key already exists, this function will return a 204 and will not overwrite it. Previous versions of Vault would overwrite the existing cert/key with new values.

Method	Path	Produces
`POST`	`/pki/root/generate/:type`	`200 application/json`

Parameters

- `type` `(string: <required>)` – Specifies the type of the root to create. If `exported`, the private key will be returned in the response; if `internal` the private key will not be returned and *cannot be retrieved later*. This is part of the request URL.
- `common_name` `(string: <required>)` – Specifies the requested CN for the certificate.
- `alt_names` `(string: "")` – Specifies the requested Subject Alternative Names, in a comma-delimited list. These can be host names or email addresses; they will be parsed into their respective fields.
- `ip_sans` `(string: "")` – Specifies the requested IP Subject Alternative Names, in a comma-delimited list.
- `uri_sans` `(string: "")` – Specifies the requested URI Subject Alternative Names, in a comma-delimited list.
- `other_sans` `(string: "")` – Specifies custom OID/UTF8-string SANs. These must match values specified on the role in `allowed_other_sans` (globbing allowed). The format is the same as OpenSSL: `<oid>;<type>:<value>` where the only current valid type is `UTF8`. This can be a comma-delimited list or a JSON string slice.
- `ttl` `(string: "")` – Specifies the requested Time To Live (after which the certificate will be expired). This cannot be larger than the engine's max (or, if not set, the system max).
- `format` `(string: "pem")` – Specifies the format for returned data. Can be `pem`, `der`, or `pem_bundle`. If `der`, the output is base64 encoded. If `pem_bundle`, the `certificate` field will contain the private key (if exported) and certificate, concatenated; if the issuing CA is not a Vault-derived self-signed root, this will be included as well.
- `private_key_format` `(string: "")` – Specifies the format for marshaling the private key. Defaults to `der` which will return either base64-encoded DER or PEM-encoded DER, depending on the value of `format`. The other option is `pkcs8` which will return the key marshalled as PEM-encoded PKCS8.
- `key_type` `(string: "rsa")` – Specifies the desired key type; must be `rsa` or `ec`.
- `key_bits` `(int: 2048)` – Specifies the number of bits to use. Must be changed to a valid value if the `key_type` is `ec`.
- `max_path_length` `(int: -1)` – Specifies the maximum path length to encode in the generated certificate. `-1` means no limit. Unless the signing certificate has a maximum path length set, in which case the path length is set to one less than that of the signing certificate. A limit of `0` means a literal path length of zero.
- `exclude_cn_from_sans` `(bool: false)` – If set, the given `common_name` will not be included in DNS or Email Subject Alternate Names (as appropriate). Useful if the CN is not a hostname or email address, but is instead some human-readable identifier.
- `permitted_dns_domains` `(string: "")` – A comma separated string (or, string array) containing DNS domains for which certificates are allowed to be issued or signed by this CA certificate. Note that subdomains are allowed, as per RFC.

- `ou` `(string: "")` – Specifies the OU (OrganizationalUnit) values in the subject field of the resulting certificate. This is a comma-separated string or JSON array.
- `organization` `(string: "")` – Specifies the O (Organization) values in the subject field of the resulting certificate. This is a comma-separated string or JSON array.
- `country` `(string: "")` – Specifies the C (Country) values in the subject field of the resulting certificate. This is a comma-separated string or JSON array.
- `locality` `(string: "")` – Specifies the L (Locality) values in the subject field of the resulting certificate. This is a comma-separated string or JSON array.
- `province` `(string: "")` – Specifies the ST (Province) values in the subject field of the resulting certificate. This is a comma-separated string or JSON array.
- `street_address` `(string: "")` – Specifies the Street Address values in the subject field of the resulting certificate. This is a comma-separated string or JSON array.
- `postal_code` `(string: "")` – Specifies the Postal Code values in the subject field of the resulting certificate. This is a comma-separated string or JSON array.

Sample Payload

```
{
  "common_name": "example.com"
}
```

Sample Request

```
$ curl \
    --header "X-Vault-Token: ..." \
    --request POST \
    --data @payload.json \
    http://127.0.0.1:8200/v1/pki/root/generate/internal
```

Sample Response

```
{
  "lease_id": "",
  "lease_duration": 0,
  "renewable": false,
  "data": {
    "certificate": "-----BEGIN CERTIFICATE-----\nMIIDzDCCAragAwIBAgIUOd0ukLcjH43TfTHFG9qE0FtlMVgwCwYJKoZIhvcNAQEL\n...\numkqeY
    "issuing_ca": "-----BEGIN CERTIFICATE-----\nMIIDzDCCAragAwIBAgIUOd0ukLcjH43TfTHFG9qE0FtlMVgwCwYJKoZIhvcNAQEL\n...\numkqeYe
    "serial": "39:dd:2e:90:b7:23:1f:8d:d3:7d:31:c5:1b:da:84:d0:5b:65:31:58"
  },
  "auth": null
}
```

Delete Root

This endpoint deletes the current CA key (the old CA certificate will still be accessible for reading until a new certificate/key are generated or uploaded). *This endpoint requires sudo/root privileges.*

Method	Path	Produces
`DELETE`	`/pki/root`	`204 (empty body)`

Sample Request

```
$ curl \
    --header "X-Vault-Token: ..." \
    --request DELETE \
    http://127.0.0.1:8200/v1/pki/root
```

Sign Intermediate

This endpoint uses the configured CA certificate to issue a certificate with appropriate values for acting as an intermediate CA. Distribution points use the values set via `config/urls`. Values set in the CSR are ignored unless `use_csr_values` is set to true, in which case the values from the CSR are used verbatim.

Method	Path	Produces
`POST`	`/pki/root/sign-intermediate`	`200 application/json`

Parameters

- `csr` `(string: <required>)` – Specifies the PEM-encoded CSR.
- `common_name` `(string: <required>)` – Specifies the requested CN for the certificate.
- `alt_names` `(string: "")` – Specifies the requested Subject Alternative Names, in a comma-delimited list. These can be host names or email addresses; they will be parsed into their respective fields.
- `ip_sans` `(string: "")` – Specifies the requested IP Subject Alternative Names, in a comma-delimited list.
- `uri_sans` `(string: "")` – Specifies the requested URI Subject Alternative Names, in a comma-delimited list.
- `other_sans` `(string: "")` – Specifies custom OID/UTF8-string SANs. These must match values specified on the role in `allowed_other_sans` (globbing allowed). The format is the same as OpenSSL: `<oid>;<type>:<value>` where the only current valid type is `UTF8`. This can be a comma-delimited list or a JSON string slice.
- `ttl` `(string: "")` – Specifies the requested Time To Live (after which the certificate will be expired). This cannot be larger than the engine's max (or, if not set, the system max). However, this can be after the expiration of the signing CA.

- `format` `(string: "pem")` – Specifies the format for returned data. Can be `pem`, `der`, or `pem_bundle`. If `der`, the output is base64 encoded. If `pem_bundle`, the `certificate` field will contain the certificate and, if the issuing CA is not a Vault-derived self-signed root, it will be concatenated with the certificate.
- `max_path_length` `(int: -1)` – Specifies the maximum path length to encode in the generated certificate. `-1`, means no limit, unless the signing certificate has a maximum path length set, in which case the path length is set to one less than that of the signing certificate. A limit of `0` means a literal path length of zero.
- `exclude_cn_from_sans` `(string: "")` – Specifies the given `common_name` will not be included in DNS or Email Subject Alternate Names (as appropriate). Useful if the CN is not a hostname or email address, but is instead some human-readable identifier.
- `use_csr_values` `(bool: false)` – If set to `true`, then: 1) Subject information, including names and alternate names, will be preserved from the CSR rather than using the values provided in the other parameters to this path; 2) Any key usages (for instance, non-repudiation) requested in the CSR will be added to the basic set of key usages used for CA certs signed by this path; 3) Extensions requested in the CSR will be copied into the issued certificate.
- `permitted_dns_domains` `(string: "")` – A comma separated string (or, string array) containing DNS domains for which certificates are allowed to be issued or signed by this CA certificate. Supports subdomains via a `.` in front of the domain, as per RFC.
- `ou` `(string: "")` – Specifies the OU (OrganizationalUnit) values in the subject field of the resulting certificate. This is a comma-separated string or JSON array.
- `organization` `(string: "")` – Specifies the O (Organization) values in the subject field of the resulting certificate. This is a comma-separated string or JSON array.
- `country` `(string: "")` – Specifies the C (Country) values in the subject field of the resulting certificate. This is a comma-separated string or JSON array.
- `locality` `(string: "")` – Specifies the L (Locality) values in the subject field of the resulting certificate. This is a comma-separated string or JSON array.
- `province` `(string: "")` – Specifies the ST (Province) values in the subject field of the resulting certificate. This is a comma-separated string or JSON array.
- `street_address` `(string: "")` – Specifies the Street Address values in the subject field of the resulting certificate. This is a comma-separated string or JSON array.
- `postal_code` `(string: "")` – Specifies the Postal Code values in the subject field of the resulting certificate. This is a comma-separated string or JSON array.

Sample Payload

```
{
  "csr": "...",
  "common_name": "example.com"
}
```

Sample Request

```
$ curl \
    --header "X-Vault-Token: ..." \
    --request POST \
    --data @payload.json \
    http://127.0.0.1:8200/v1/pki/root/sign-intermediate
```

Sample Response

```
{
  "lease_id": "",
  "renewable": false,
  "lease_duration": 0,
  "data": {
    "certificate": "-----BEGIN CERTIFICATE-----\nMIIDzDCCAragAwIBAgIUOd0ukLcjH43TfTHFG9qE0FtlMVgwCwYJKoZIhvcNAQEL\n...\numkqe
    "issuing_ca": "-----BEGIN CERTIFICATE-----\nMIIDUTCCAjmgAwIBAgIJAKM+z4MSfw2mMA0GCSqGSIb3DQEBCwUAMBsxGTAXBgNV\n...\nG/7g4k
    "ca_chain": ["-----BEGIN CERTIFICATE-----\nMIIDUTCCAjmgAwIBAgIJAKM+z4MSfw2mMA0GCSqGSIb3DQEBCwUAMBsxGTAXBgNV\n...\nG/7g4ko
    "serial": "39:dd:2e:90:b7:23:1f:8d:d3:7d:31:c5:1b:da:84:d0:5b:65:31:58"
  },
  "auth": null
}
```

Sign Self-Issued

This endpoint uses the configured CA certificate to sign a self-issued certificate (which will usually be a self-signed certificate as well).

This is an extremely privileged endpoint. The given certificate will be signed as-is with only minimal validation performed (is it a CA cert, and is it actually self-issued). The only values that will be changed will be the authority key ID, the issuer DN, and, if set, any distribution points.

This is generally only needed for root certificate rolling in cases where you don't want/can't get access to a CSR (such as if it's a root stored in Vault where the key is not exposed). If you don't know whether you need this endpoint, you most likely should be using a different endpoint (such as `sign-intermediate`).

This endpoint requires `sudo` capability.

Method	Path	Produces
`POST`	`/pki/root/sign-self-issued`	`200 application/json`

Parameters

- `certificate` `(string: <required>)` – Specifies the PEM-encoded self-issued certificate.

Sample Payload

```
{
  "certificate": "..."
}
```

Sample Request

```
$ curl \
    --header "X-Vault-Token: ..." \
    --request POST \
    --data @payload.json \
    http://127.0.0.1:8200/v1/pki/root/sign-self-issued
```

Sample Response

```
{
  "lease_id": "",
  "renewable": false,
  "lease_duration": 0,
  "data": {
    "certificate": "-----BEGIN CERTIFICATE-----\nMIIDzDCCAragAwIBAgIUOd0ukLcjH43TfTHFG9qE0FtlMVgwCwYJKoZIhvcNAQEL\n...\numkqeY
    "issuing_ca": "-----BEGIN CERTIFICATE-----\nMIIDUTCCAjmgAwIBAgIJAKM+z4MSfw2mMA0GCSqGSIb3DQEBCwUAMBsxGTAXBgNV\n...\nG/7g4kc
  },
  "auth": null
}
```

Sign Certificate

This endpoint signs a new certificate based upon the provided CSR and the supplied parameters, subject to the restrictions contained in the role named in the endpoint. The issuing CA certificate is returned as well, so that only the root CA need be in a client's trust store.

Method	Path	Produces
POST	/pki/sign/:name	200 application/json

Parameters

- `csr` `(string: <required>)` – Specifies the PEM-encoded CSR.
- `common_name` `(string: <required>)` – Specifies the requested CN for the certificate. If the CN is allowed by role policy, it will be issued.

- `alt_names` `(string: "")` – Specifies the requested Subject Alternative Names, in a comma-delimited list. These can be host names or email addresses; they will be parsed into their respective fields. If any requested names do not match role policy, the entire request will be denied.

- `other_sans` `(string: "")` – Specifies custom OID/UTF8-string SANs. These must match values specified on the role in `allowed_other_sans` (globbing allowed). The format is the same as OpenSSL: `<oid>;<type>:<value>` where the only current valid type is `UTF8`. This can be a comma-delimited list or a JSON string slice.

- `ip_sans` `(string: "")` – Specifies the requested IP Subject Alternative Names, in a comma-delimited list. Only valid if the role allows IP SANs (which is the default).

- `uri_sans` `(string: "")` – Specifies the requested URI Subject Alternative Names, in a comma-delimited list. If any requested URIs do not match role policy, the entire request will be denied.

- `ttl` `(string: "")` – Specifies the requested Time To Live. Cannot be greater than the role's `max_ttl` value. If not provided, the role's `ttl` value will be used. Note that the role values default to system values if not explicitly set.

- `format` `(string: "pem")` – Specifies the format for returned data. Can be `pem`, `der`, or `pem_bundle`. If `der`, the output is base64 encoded. If `pem_bundle`, the `certificate` field will contain the certificate and, if the issuing CA is not a Vault-derived self-signed root, it will be concatenated with the certificate.

- `exclude_cn_from_sans` `(bool: false)` – If set, the given `common_name` will not be included in DNS or Email Subject Alternate Names (as appropriate). Useful if the CN is not a hostname or email address, but is instead some human-readable identifier.

Sample Payload

```
{
  "csr": "...",
  "common_name": "example.com"
}
```

Sample Response

```
{
  "lease_id": "pki/sign/test/7ad6cfa5-f04f-c62a-d477-f33210475d05",
  "renewable": false,
  "lease_duration": 21600,
  "data": {
    "certificate": "-----BEGIN CERTIFICATE-----\nMIIDzDCCAragAwIBAgIUOd0ukLcjH43TfTHFG9qE0FtlMVgwCwYJKoZIhvcNAQEL\n...\numkqe
    "issuing_ca": "-----BEGIN CERTIFICATE-----\nMIIDUTCCAjmgAwIBAgIJAKM+z4MSfw2mMA0GCSqGSIb3DQEBCwUAMBsxGTAXBgNV\n...\nG/7g4k
    "ca_chain": ["-----BEGIN CERTIFICATE-----\nMIIDUTCCAjmgAwIBAgIJAKM+z4MSfw2mMA0GCSqGSIb3DQEBCwUAMBsxGTAXBgNV\n...\nG/7g4ko
    "serial": "39:dd:2e:90:b7:23:1f:8d:d3:7d:31:c5:1b:da:84:d0:5b:65:31:58"
  },
  "auth": null
}
```

Sign Verbatim

This endpoint signs a new certificate based upon the provided CSR. Values are taken verbatim from the CSR; the *only* restriction is that this endpoint will refuse to issue an intermediate CA certificate (see the `/pki/root/sign-intermediate` endpoint for that functionality.)

This is a potentially dangerous endpoint and only highly trusted users should have access.

Method	Path	Produces
`POST`	`/pki/sign-verbatim(/:name)`	`200 application/json`

Parameters

- `name` `(string: "")` - Specifies a role. If set, the following parameters from the role will have effect: `ttl` , `max_ttl` , `generate_lease` , and `no_store` .
- `csr` `(string: <required>)` – Specifies the PEM-encoded CSR.
- `key_usage` `(list: ["DigitalSignature", "KeyAgreement", "KeyEncipherment"])` – Specifies the allowed key usage constraint on issued certificates. Valid values can be found at https://golang.org/pkg/crypto/x509/#KeyUsage - simply drop the `KeyUsage` part of the value. Values are not case-sensitive. To specify no key usage constraints, set this to an empty list.
- `ext_key_usage` `(list: [])` – Specifies the allowed extended key usage constraint on issued certificates. Valid values can be found at https://golang.org/pkg/crypto/x509/#ExtKeyUsage - simply drop the `ExtKeyUsage` part of the value. Values are not case-sensitive. To specify no key usage constraints, set this to an empty list.
- `ttl` `(string: "")` – Specifies the requested Time To Live. Cannot be greater than the engine's `max_ttl` value. If not provided, the engine's `ttl` value will be used, which defaults to system values if not explicitly set.
- `format` `(string: "pem")` – Specifies the format for returned data. Can be `pem` , `der` , or `pem_bundle` . If `der` , the output is base64 encoded. If `pem_bundle` , the `certificate` field will contain the certificate and, if the issuing CA is not a Vault-derived self-signed root, it will be concatenated with the certificate.

Sample Payload

```
{
  "csr": "..."
}
```

Sample Request

```
$ curl \
    --header "X-Vault-Token: ..." \
    --request POST \
    --data @payload.json \
    http://127.0.0.1:8200/v1/pki/sign-verbatim
```

Sample Response

```
{
  "lease_id": "pki/sign-verbatim/7ad6cfa5-f04f-c62a-d477-f33210475d05",
  "renewable": false,
  "lease_duration": 21600,
  "data": {
    "certificate": "-----BEGIN CERTIFICATE-----\nMIIDzDCCAragAwIBAgIUOd0ukLcjH43TfTHFG9qE0FtlMVgwCwYJKoZIhvcNAQEL\n...\numkqe
    "issuing_ca": "-----BEGIN CERTIFICATE-----\nMIIDUTCCAjmgAwIBAgIJAKM+z4MSfw2mMA0GCSqGSIb3DQEBCwUAMBsxGTAXBgNV\n...\nG/7g4k
    "ca_chain": ["-----BEGIN CERTIFICATE-----\nMIIDUTCCAjmgAwIBAgIJAKM+z4MSfw2mMA0GCSqGSIb3DQEBCwUAMBsxGTAXBgNV\n...\nG/7g4ko
    "serial": "39:dd:2e:90:b7:23:1f:8d:d3:7d:31:c5:1b:da:84:d0:5b:65:31:58"
  },
  "auth": null
}
```

Tidy

This endpoint allows tidying up the storage backend and/or CRL by removing certificates that have expired and are past a certain buffer period beyond their expiration time.

Method	Path	Produces
POST	/pki/tidy	200 application/json

Parameters

- `tidy_cert_store` `(bool: false)` Specifies whether to tidy up the certificate store.
- `tidy_revoked_certs` `(bool: false)` Set to true to expire all revoked and expired certificates, removing them both from the CRL and from storage. The CRL will be rotated if this causes any values to be removed.
- `safety_buffer` `(string: "")` Specifies A duration (given as an integer number of seconds or a string; defaults to `72h`) used as a safety buffer to ensure certificates are not expunged prematurely; as an example, this can keep certificates from being removed from the CRL that, due to clock skew, might still be considered valid on other hosts. For a certificate to be expunged, the time must be after the expiration time of the certificate (according to the local clock) plus the duration of `safety_buffer` .

Sample Payload

```
{
  "safety_buffer": "24h"
}
```

Sample Request

```
$ curl \
    --header "X-Vault-Token: ..." \
    --request POST \
    --data @payload.json \
    http://127.0.0.1:8200/v1/pki/tidy
```