Deckhouse Stronghold Documentation

Generated: July 07, 2026

Documentation

About the product ... 7
- Editions ... 7
- Requirements ... 9

Concepts ... 10
- Lease ... 10
- Authentication ... 11
- Response wrapping ... 13
- Policies ... 16
- Password policy ... 33
- Identity ... 42
- Token ... 47

Install ... 54
- Running on Linux OS ... 54
  - Installation ... 54
  - Configuration ... 61
  - Recover from lost quorum ... 84
- Launch in DKP ... 89
  - Release channels ... 89
  - Platform installation ... 90
    - Installation steps ... 90
      - Prepare the environment ... 90
      - Platform installation ... 91
      - Initial access configuration ... 100
  - Platform configuration ... 101
    - Node management ... 101
      - Node configuration ... 101
        - OS configuration ... 101
        - Containerd configuration ... 101
        - Node configuration ... 101
      - Node groups ... 101
      - Adding a node ... 101
      - Advanced configuration ... 101
    - Control plane ... 101

Components ... 101
Description ... 101
Overview ... 101
Etcd operations ... 101
Scheduler ... 102
Audit ... 102
Master nodes ... 102
Network management ... 102
Incoming traffic management ... 102
Outgoing traffic management ... 105
Routing management ... 108
DNS management ... 112
NTP management ... 115
Monitoring ... 116
Node monitoring ... 116
Access control ... 117
Role model ... 117
User management ... 120
Projects ... 128
Platform update ... 130
Platform update ... 130
Manual update mode ... 130
Removing the platform ... 131
Stronghold configuration ... 132
Administration ... 133
Overview ... 133
Audit ... 134
Introduction ... 134
Audit log record schema ... 138
Audit field exclusion ... 143
Audit log filtering ... 146
Backup and Restore ... 149
Overview ... 149
Save a storage snapshot ... 150
Inspect a snapshot ... 151
Restore from a snapshot ... 153

Automated snapshots ... 154
KMS and HSM ... 160
HSM support ... 160
Yandex Cloud KMS ... 165
Double encryption ... 167
KV1/KV2 replication ... 170
Introduction ... 170
Namespaces ... 170
Introduction ... 170
Cryptographic algorithms ... 170
Introduction ... 170
Extensions and Integrations ... 171
Introduction ... 171
Plugins in Standalone ... 171
Plugins in DKP ... 174
Usage ... 179
Overview ... 179
Get started ... 179
Configuring access ... 179
Creating a test secret ... 180
Auth methods ... 181
Overview ... 181
OIDC providers ... 183
Overview ... 183
Gitlab ... 183
Keycloak ... 183
Kubernetes ... 183
JWT ... 189
Approle ... 192
userpass ... 196
Token ... 201
LDAP ... 202
SAML ... 209
Kubernetes ... 215
WebAuthn ... 220
2FA / MFA ... 223
MULTIFACTOR Ldap Adapter ... 223

TOTP ... 228
Managed Keys ... 230
Overview ... 230
Secrets engines ... 233
Overview ... 233
KV Secrets engine ... 235
KV secrets engine ... 235
KV/v1 ... 235
KV/v2 ... 238
KV1/KV2 replication ... 250
PKI Secrets engine ... 250
Cubbyhole Secrets engine ... 254
SSH Secrets engine ... 255
Transit Secrets engine ... 265
TOTP Secrets engine ... 270
Kubernetes Secrets engine ... 273
LDAP Secrets engine ... 281
Identity Secrets engine ... 290
Overview ... 290
OIDC identity provider ... 291
OIDC identity tokens ... 294
Databases Secrets engine ... 298
Overview ... 298
PostgreSQL ... 302
MySQL ... 304
Stronghold agent ... 307
Overview ... 307
Use cases ... 309
Key features ... 309
Main settings ... 309
Launch and management ... 309
Reference ... 310
API ... 310
auth ... 310
secrets ... 411
identity ... 753
system ... 801

Console utilities 910
FAQ **911**
Release notes **912**

About the product

Editions

Deckhouse Stronghold is available as Community Edition (CE) and Enterprise Edition (EE).

Deckhouse Stronghold CE is available for use in any of the Deckhouse Kubernetes Platform editions.

Deckhouse Stronghold EE is licensed separately and available for use in any **commercial edition** of DKP.

The table below provides a brief comparison of the Deckhouse Stronghold editions, listing their main features and details:

Feature	CE	EE
Secure management of the secret lifecycle (storage, creation, delivery, revocation, and rotation)	[✓]	[✓]
Support of IaC automation tools (Ansible, Terraform)	[✓]	[✓]
Support of authentication methods	JWT, OIDC, Kubernetes, LDAP, Token, WebAuthn	JWT, OIDC, Kubernetes, LDAP, Token, WebAuthn, SAML
Support of KV, Kubernetes, Database, SSH, and PKI secret engines	[✓]	[✓]
Deploying to an air-gapped environment	[✓]	[✓]
Web interface	[✓]	[✓]
Role and access policy management through a web interface	[✗]	[✓]
Support for namespaces	[✗]	[✓]
Built-in automatic vault unsealing without requiring any external services or KMS	[✗]	[✓]
HA configurations	[✓]	[✓]
Trans-cluster data replication	[✗]	KV1/KV2
Automatic backup creation on a schedule	[✗]	[✓]
Audit logging support	[✗]	[✓]
Managed Keys	[✗]	[✓]
GOST algorithm support for PKI/Transit	[✓]	[✓]
Delivered as a standalone executable file	[✗]	[✓]
Can be launched in DKP CE	[✓]	[✗]

Requirements

System requirements

Since the way each customer uses the secrets storage may vary, the following recommendations should be considered as a starting point. The scope of required resources depends on operations performed by the Stronghold cluster.

Below are the two main cluster types considered depending on their purpose:

- **Small clusters**: Suitable for initial deployments, as well as development and testing environments.
- **Large clusters**: Intended for production environments with consistently high workloads. This may involve a large number of transactions, secrets, or a combination of both conditions.

	Small cluster	**Large cluster**
CPU	4–8 cores	8–16 cores
Memory	8–16 GB	16–32 GB
Disk input/output	3000+ IOPS	3000+ IOPS
Disk input/output	70+ MB/s	200+ MB/s

Depending on the expected number and type of operations, consider the following requirements:

Operation	**4 cores**	**16 cores**
Authorization (token retrieval)	Up to 20 OPS	Up to 100 OPS
Reading a key (up to 1 KB)	Up to 500 OPS	Up to 7000 OPS
Writing a key	Up to 30 OPS	Up to 150 OPS

Supported OS

Linux distribution	**Supported versions**
CentOS	7, 8, 9
Debian	10, 11, 12
Ubuntu	18.04, 20.04, 22.04, 24.04

Concepts

Lease

Lease, renewal, and revoke

For every dynamic secret and authentication service, Stronghold creates a *lease*, which includes metadata containing information such as duration, renewability, and more. Stronghold guarantees that the data will be valid for the given duration (Time to Live, TTL). Once the lease is expired, Stronghold can automatically revoke the data, and the consumer of the secret can no longer be certain that it is valid.

Consumers of secrets have to check in with Stronghold routinely to either renew the lease (if allowed) or request a replacement secret. This improves the value of Stronghold audit logs and significantly simplifies the key replacement process.

All dynamic secrets in Stronghold are required to have a lease. Even if the data is meant to be valid “forever”, a lease is required to force the consumer to check in routinely.

In addition to renewals, a lease can be *revoked*. When a lease is revoked, it invalidates that secret immediately and prevents any further renewals.

The revocation can be done manually via the API, via the `d8 stronghold lease revoke` CLI command, via the user interface under the “Access” tab, or automatically by Stronghold. When a lease is expired, Stronghold automatically revokes it. When a token is revoked, Stronghold revokes all leases that were created using it.

> The Key/Value backend, which stores arbitrary secrets, doesn’t issue leases but sometimes returns a lease duration. For details, refer to the `kv` secrets engine documentation.

Lease IDs

When reading a dynamic secret (for example, using the `d8 stronghold read` command), Stronghold always returns a `lease_id`. This ID can be used in commands such as `d8 stronghold lease renew` and `d8 stronghold lease revoke` to manage the lease of a secret.

Lease duration and renewal

Lease duration is returned along with the lease ID as a Time To Live (TTL) value, time in seconds for which the lease is valid. A consumer of this secret must renew the lease within that timeframe.

When renewing the lease, the user can request a specific amount of time they want remaining on the lease, which is called the `increment`. This increment to the lease duration won’t be added at the end of

the current TTL but rather at the request time. For example, the command `d8 stronghold lease renew -increment=3600 my-lease-id` would request that the TTL of the lease be adjusted to 1 hour (3600 seconds). Having the increment be rooted at the current time instead of the end of the lease lets users increase or reduce the length of leases if they don't need a secret for the full possible lease period.

The requested `increment` is completely advisory. The backend in charge of the secret can choose to completely ignore it. For most secrets, the backend does its best to respect the `increment`, but often limits it to ensure renewals every so often.

The return value of renewals should be carefully inspected to determine what the new lease TTL is.

Prefix-based revocation

In addition to revoking a single secret, users with proper access control can revoke multiple secrets based on their lease ID prefix.

The lease ID prefixes always contain the path where the secret was requested from. This lets you revoke groups of secrets. For example, to revoke all Userpass logins, it would be enough to run `d8 stronghold lease revoke -prefix auth/userpass/`.

This can be useful if there is an intrusion within a system. All secrets of a specific backend or a certain configuration can be revoked quickly.

Authentication

Authentication in Stronghold is the process by which user or machine supplied information is verified against an internal or external system. Stronghold supports multiple auth methods including Userpass, LDAP, AppRole, and more. Each auth method has a specific use case.

Before interacting with Stronghold, each client must *authenticate* using one of the supported authentication methods. Upon successful authentication, a token is generated, which is conceptually similar to a session ID on a website. The token may have attached policy, which is mapped at the authentication time. For details on this process, refer to Policies.

Authentication methods

Stronghold supports a number of authentication methods. Some backends are targeted toward users while others are targeted toward machines. Most authentication backends must be enabled before use.

To enable an authentication method, run the following command:

```
d8 stronghold write sys/auth/my-auth type=userpass
```

This enables the `userpass` authentication method at the `my-auth` path. This authentication will be accessible at the `my-auth` path. Often you will see authentications at the same path as their name, but this is not a requirement.

To learn more about this authentication, use the built-in `path-help` command:

```
d8 stronghold path-help auth/my-auth
```

Stronghold supports multiple authentication methods simultaneously, and you can even mount the same type of authentication method at different paths. Only one authentication is required to gain access to Stronghold, and it is not currently possible to force a user through multiple auth methods to gain access, although some backends do support the multifactor authentication (MFA).

Tokens

Authentication works by verifying your identity and then generating a token to associate with that identity.

For example, even though you may authenticate using DEX, Stronghold generates a unique access token for you to use for future requests. The CLI automatically attaches this token to requests, but if you're using the API you'll have to do this manually.

This token given for authentication with any backend can also be used with the full set of token commands, such as creating new sub-tokens, revoking tokens, and renewing tokens. For details, refer to Tokens.

Authentication

Via the CLI

To authenticate with the CLI, `d8 stronghold login` is used. This supports many of the built-in authentication methods.

For example, to authenticate with OIDC, run the following command:

```
d8 stronghold login -method=oidc
```

After authenticating, you will be logged in. The CLI command will also output your raw token. This token is used for revocation and renewal. As the user logging in, the primary use case of the token is renewal, which is covered below in Authentication leases.

To determine what variables are needed for an auth method, supply the `-method` flag without any additional arguments and help will be shown.

If you're using a method that isn't supported via the CLI, then the API must be used.

Via the API

API authentication is generally used for machine authentication. Each authentication method implements its own login endpoint.

To find the proper endpoint, use the following command:

```
d8 stronghold path-help
```

Authentication leases

Just like secrets, identities have leases associated with them. This means that you must reauthenticate after the given lease period to continue accessing Stronghold.

To set the lease associated with an identity, reference the documentation for the specific auth method in use. It is specific to each backend how leasing is implemented.

And just like secrets, identities can be renewed without having to completely reauthenticate. To renew it, use the following command, specifying the token associated with your identity:

```
d8 stronghold token renew <token>
```

Response wrapping

In many deployment scenarios of Deckhouse Stronghold, clients interact directly with Stronghold and use the returned secrets. However, in some cases, it may be more appropriate to separate privileges so that one trusted party interacts with most of the Stronghold API and then passes the secrets to the final consumer.

The more intermediaries involved in secret transmission, the higher the risk of accidental disclosure, especially if the secret is transmitted in plaintext. For example, you may need to deliver a private TLS key to a machine where, due to security policy, the decryption key can't be stored in persistent storage. In this case, encrypting the key before transfer is not possible.

To address such scenarios, Stronghold provides the *response wrapping* feature. Instead of returning the response directly to the HTTP client, Stronghold stores it in the `cubbyhole`. Access to the stored content is granted only through a one-time token that Stronghold returns to the client.

From a logical perspective, the response is wrapped in a token that must be unwrapped to access the data. From a functional perspective, the token authorizes access to Stronghold's "key holder" and allows the data to be decrypted.

This method provides a reliable mechanism for secure information exchange across various environments. Response wrapping solves three key tasks:

- **Hiding sensitive information**. The value transmitted over the network is not the secret itself but a token to access it. Even if intercepted or logged, it does not contain confidential information.
- **Abuse detection**. A token can only be unwrapped by a single party. If a client receives a token that cannot be unwrapped, it is a reason to initiate a security incident investigation. Before unwrapping, the client can verify the token's origin through Stronghold.
- **Expiration enforcement**. A token has its own time-to-live (TTL), separate from the TTL of the secret (and usually much shorter). If the client does not unwrap the token in time, it expires.

Response wrapping tokens

When wrapping is applied, Stronghold does not return the original API response directly. Instead, the client receives the following token metadata:

- TTL — token lifetime.
- Token — token value.
- Creation time — time when the token was generated.
- Creation path — API endpoint that triggered the response generation.
- Wrapped token ID — indicates the wrapped token (for example, during authentication or token creation). This is useful for orchestration systems (such as Nomad), as it allows token lifetime management without disclosing the token.

Currently, Stronghold does not support signing of response wrapping tokens, since it does not provide significant additional protection. If the server endpoint is correct, token validation is performed by interacting with Stronghold itself. A signed token would not eliminate the need to verify it with the server, because the token does not contain the sensitive data itself—it is only a mechanism to access it. Therefore, Stronghold will not return the data without confirming the token's validity.

Even if an attacker redirects a client to a fake server, they can substitute the public signing key. In theory, you could cache a previously valid key, but that would also require caching the previously valid address (in most cases, the Stronghold address will not change or will be set via service discovery).

Thus, Stronghold relies on the fact that the token does not contain sensitive data and therefore does not require signing.

Operations with wrapping tokens

The following operations are available under the `sys/wrapping/` path:

- **Lookup** (`sys/wrapping/lookup`): Returns the token's creation time, path, and TTL. The path does not require authentication. The token holder can always require its properties via this endpoint.
- **Unwrap** (`sys/wrapping/unwrap`): Unwraps the token, returning the stored response in its original API format.
- **Rewrap** (`sys/wrapping/rewrap`): Rewraps already wrapped data into a new token. This is useful for secrets with long lifetimes. For example, an organization may want (or be required by security policy) to have the backend `pki` root CA key returned in a long-lived wrapping token—ensuring the key is never exposed (which can be verified via token lookup)—but still needs access to the key to sign a CRL if the `pki` mount is changed or lost. Security policies often require secret rotation, and this operation enables it without risk of exposure.
- **Wrap** (`sys/wrapping/wrap`): Returns data wrapped in a token.

> Even if access to this path is restricted, wrapping can still be performed through other Stronghold API methods.

Creating a wrapping token

Response wrapping is performed per request and is triggered by specifying the desired TTL. The TTL is set by the client using the `X-Vault-Wrap-TTL` header and can be either an integer (in seconds) or a string (`15s` , `20m` , `25h`). In the Stronghold CLI, this is done with the `-wrap-ttl` option. In the Go API, the `SetWrappingLookupFunc` function is used. It instructs the API under which conditions wrapping should be requested by matching the operation and path with the desired TTL.

Wrapping process:

1. The original HTTP response is serialized.
2. A one-time token is generated with the TTL set by the client.
3. The response is stored in the `cubbyhole` associated with the token.
4. A new response is generated with additional fields containing token information (ID, TTL, and path).
5. The new response is returned to the client.

The minimum and maximum TTL of wrapping tokens is controlled by policies.

Verifying a wrapping token

To reduce abuse risks, it is recommended to validate wrapping tokens using the following procedure:

1. If the client expected a token but did not receive one, it may have been intercepted by an attacker. Start an incident investigation.
2. Perform a lookup on the token. If the token has expired or been revoked, this does not always mean a leak (for example, the client started too late). However, investigation is still required. Using the Stronghold audit log, check whether the token was unwrapped.
3. Compare the token's creation path with the expected one. For example, if you expected a TLS key or certificate after unwrapping, the path should likely be `pki/issue/...` . A mismatch may indicate token substitution (most likely if the path starts with `cubbyhole` or `sys/wrapping/wrap`).

   > Pay special attention to the `kv` secrets engine. If you expect the secret to come from `secret/foo` , but an attacker passes a token with the path `secret/bar` , simply checking the `secret/` prefix is insufficient.

4. After verifying the prefix, unwrap the token. If unwrapping fails, initiate an incident investigation.

By following these steps, you can be sure that only the intended client has seen the data inside the wrapping token, and any attempt at substitution or interception will be detected.

Policies

Policies provide a declarative way to grant or forbid access to certain paths and operations in Stronghold. This section discusses policy workflows and syntaxes.

Policies are deny by default, so an empty policy grants no permission in the system.

Policy-authentication workflow

Before a human or machine can gain access, an administrator must configure Stronghold with an auth method. Authentication is the process by which human or machine-supplied information is verified against an internal or external system.

Consider the following diagram, which illustrates the steps a security team would take to configure Stronghold to authenticate using a corporate LDAP or Active Directory installation. Even though this example uses LDAP, the concept applies to all auth methods.

1. The security team configures Stronghold to connect to an auth method. This configuration varies by auth method. In the case of LDAP, Stronghold needs to know the address of the LDAP server and whether to connect using TLS. It is important to note that Stronghold does not store a copy of the LDAP database — Stronghold will delegate the authentication to the auth method.
2. The security team authors a policy (or uses an existing policy) which grants access to paths in Stronghold. Policies are written in HCL in your editor of preference and saved to disk.
3. The policy's contents are uploaded and stored in Stronghold and referenced by name. You can think of the policy's name as a pointer or symlink to its set of rules.
4. Most importantly, the security team maps data in the auth method to a policy. For example, the security team might create mappings like:
    - Members of the OU group "dev" map to the Stronghold policy named "readonly-dev".
    - Members of the OU group "ops" map to the Stronghold policies "admin" and "auditor".

Now Stronghold has an internal mapping between a backend authentication system and internal policy. When a user authenticates to Stronghold, the actual authentication is delegated to the auth method.

For a user, the authentication flow looks like:

1. A user attempts to authenticate to Stronghold using their LDAP credentials, providing Stronghold with their LDAP username and password.
2. Stronghold establishes a connection to LDAP and asks the LDAP server to verify the given credentials. Assuming this is successful, the LDAP server returns the information about the user, including the OU groups.
3. Stronghold maps the result from the LDAP server to policies inside Stronghold using the mapping configured by the security team in the previous section. Stronghold then generates a token and attaches the matching policies.

4. Stronghold returns the token to the user. This token has the correct policies assigned, as dictated by the mapping configuration that was setup by the security team in advance.
5. The user then uses this Stronghold token for future operations. If the user performs the authentication steps again, they will get a new token. The token will have the same permissions, but the actual token will be different. Authenticating a second time does not invalidate the original token.

Policy syntax

Policies are written in HCL or JSON and describe which paths in Stronghold a user or machine is allowed to access.

Here is a simple policy which grants read capabilities to the KV/v1 path `secret/foo`:

```
path "secret/foo" {
  capabilities = ["read"]
}
```

When this policy is assigned to a token, the token can read from `secret/foo`. However, the token cannot update or delete `secret/foo`, since the capabilities do not allow it. Because policies are deny by default, the token would have no other access in Stronghold.

Here is a more detailed policy, and it is documented inline:

```
# This section grants all access on "secret/*".
# Further restrictions can be applied to this broad policy, as shown below.
path "secret/*" {
  capabilities = ["create", "read", "update", "patch", "delete", "list"]
}

# Even though we allowed "secret/*", this line explicitly denies "secret/super-
secret".
# This takes precedence.
path "secret/super-secret" {
  capabilities = ["deny"]
}

# Policies can also specify allowed, disallowed, and required parameters.
# Here the key "secret/restricted" can only contain "foo" (any value) and "bar" ("zip"
or "zap").
path "secret/restricted" {
  capabilities = ["create"]
  allowed_parameters = {
    "foo" = []
    "bar" = ["zip", "zap"]
  }
}
```

Policies use path-based matching to test the set of capabilities against a request. A policy path may specify an exact path to match, or it could specify a glob pattern which instructs Stronghold to use a prefix match:

```
# Permit reading only "secret/foo".
# An attached token cannot read "secret/food" or "secret/foo/bar".
path "secret/foo" {
  capabilities = ["read"]
}

# Permit reading everything under "secret/bar".
# An attached token could read "secret/bar/zip", "secret/bar/zip/zap", but not "secret
/bars/zip".
path "secret/bar/*" {
  capabilities = ["read"]
}

# Permit reading everything prefixed with "zip-".
# An attached token could read "secret/zip-zap" or "secret/zip-zap/zong", but not "sec
ret/zip/zap".
path "secret/zip-*" {
  capabilities = ["read"]
}
```

In addition, a `+` (plus) can be used to denote any number of characters bounded within a single path segment:

```
# Permit reading the "teamb" path under any top-level path under "secret/".
path "secret/+/teamb" {
  capabilities = ["read"]
}

# Permit reading "secret/foo/bar/teamb", "secret/bar/foo/teamb", etc.
path "secret/+/+/teamb" {
  capabilities = ["read"]
}
```

Stronghold's architecture is similar to a filesystem. Every action in Stronghold has a corresponding path and capability. Even Stronghold's internal core configuration endpoints live under the `sys/` path. Policies define access to these paths and capabilities, which controls a token's access to credentials in Stronghold.

The policy rules that Stronghold applies are determined by the most-specific match available, using the priority rules described below. This may be an exact match or the longest-prefix match of a glob. If the same pattern appears in multiple policies, we take the union of the capabilities. If different patterns appear in the applicable policies, we take only the highest-priority match from those policies.

This means if you define a policy for `secret/foo*`, the policy would also match `secret/foobar`.

> The glob character referred to in this documentation is the asterisk (`*`). It is not a regular expression and is only supported as the last character of the path!

When providing list capability, it is important to note that since listing always operates on a prefix, policies must operate on a prefix because Stronghold will sanitize request paths to be prefixes.

Capabilities

Each path must define one or more capabilities which provide fine-grained control over permitted (or denied) operations. As shown in the examples above, capabilities are always specified as a list of strings, even if there is only one capability.

To determine the capabilities needed to perform a specific operation, the `-output-policy` flag can be added to the CLI subcommand.

The list of capabilities includes the following:

- `create` (POST/PUT): Allows creating data at the given path. Very few parts of Stronghold distinguish between create and update, so most operations require both create and update capabilities.
- `read` (GET): Allows reading the data at the given path.

- `update` (POST/PUT): Allows changing the data at the given path. In most parts of Stronghold, this implicitly includes the ability to create the initial value at the path.
- `patch` (PATCH): Allows partial updates to the data at a given path.
- `delete` (DELETE): Allows deleting the data at the given path.
- `list` (LIST): Allows listing values at the given path. Note that the keys returned by a list operation are not filtered by policies. Do not encode sensitive information in key names. Not all backends support listing.

In the list above, the associated HTTP verbs are shown in parenthesis next to the capability. When authoring policy, it is usually helpful to look at the HTTP API documentation for the paths and HTTP verbs and map them back onto capabilities. While the mapping is not strictly 1:1, they are often very similarly matched.

In addition to the standard set, there are some capabilities that do not map to HTTP verbs:

- `sudo` : Allows access to paths that are root-protected. Tokens are not permitted to interact with these paths unless they have the `sudo` capability (in addition to the other necessary capabilities for performing an operation against that path, such as read or delete).

  For example, modifying the audit log backends requires a token with sudo privileges.

- `deny` : Disallows access. This always takes precedence regardless of any other defined capabilities, including sudo.

> Capabilities usually map to the HTTP verb, and not the underlying action taken. This can be a common source of confusion. Generating database credentials creates database credentials, but the HTTP request is a GET which corresponds to a `read` capability. Thus, to grant access to generate database credentials, the policy would grant read access on the appropriate path.

Templated policies

The policy syntax allows for doing variable replacement in some policy strings with values available to the token. Currently identity information can be injected, and currently the path keys in policies allow injection.

Parameters

Name	Description
`identity.entity.id`	Entity's ID
`identity.entity.name`	Entity's name
`identity.entity.metadata.<metadata key>`	Metadata associated with the entity for the given key
`identity.entity.aliases.<mount accessor>.id`	Entity alias ID for the given mount
`identity.entity.aliases.<mount accessor>.name`	Entity alias name for the given mount
`identity.entity.aliases.<mount accessor>.metadata.<metadata key>`	Metadata associated with the alias for the given mount and metadata key
`identity.entity.aliases.<mount accessor>.custom_metadata.<custom_metadata key>`	Custom metadata associated with the alias for the given mount and custom metadata key
`identity.groups.ids.<group id>.name`	Group name for the given group ID
`identity.groups.names.<group name>.id`	Group ID for the given group name
`identity.groups.ids.<group id>.metadata.<metadata key>`	Metadata associated with the group for the given key
`identity.groups.names.<group name>.metadata.<metadata key>`	Metadata associated with the group for the given key

Examples

The following policy creates a section of the KVv2 Secret Engine to a specific user

```
path "secret/data/{{identity.entity.id}}/*" {
  capabilities = ["create", "update", "patch", "read", "delete"]
}

path "secret/metadata/{{identity.entity.id}}/*" {
  capabilities = ["list"]
}
```

If you wanted to create a shared section of KV that is associated with entities that are in a group.

```
# In the example below, the group ID maps a group and the path
path "secret/data/groups/
{{identity.groups.ids.fb036ebc-2f62-4124-9503-42aa7A869741.name}}/*" {
  capabilities = ["create", "update", "patch", "read", "delete"]
}

path "secret/metadata/groups/
{{identity.groups.ids.fb036ebc-2f62-4124-9503-42aa7A869741.name}}/*" {
  capabilities = ["list"]
}
```

> When developing templated policies, use IDs wherever possible. Each ID is unique to the user, whereas names can change over time and can be reused. This ensures that if a given user or group name is changed, the policy will be mapped to the intended entity or group.

If you want to use the metadata associated with an authentication plugin in your templates, you will need to get its mount accessor and access it via the aliases key.

You can get the mount accessor value using the following command:

```
d8 stronghold auth list
```

Example output:

```
Path           Type         Accessor                    Description
----           ----         --------                    -----------
kubernetes/    kubernetes   auth_kubernetes_xxxx        n/a
token/         token        auth_token_yyyy             token based credentials
```

The following templated policy allow to read the path associated with the Kubernetes service account namespace of the identity:

```
path "secret/data/
{{identity.entity.aliases.auth_kubernetes_xxxx.metadata.service_account_namespace}}/*"
{
  capabilities = ["read"]
}
```

Fine-grained control

In addition to the standard set of capabilities, Stronghold offers finer-grained control over permissions at a given path. The capabilities associated with a path take precedence over permissions on parameters.

Parameter constraints

- The use of globs (`*`) may result in surprising or unexpected behavior.
- The `allowed_parameters` , `denied_parameters` , and `required_parameters` fields are not supported for policies used with the KV/v2 secrets engine.

Policies can take into account HTTP request parameters to further constrain requests, using the following options:

- `required_parameters` : A list of parameters that must be specified.

```
# This requires the user to create "secret/profile" with a parameter/key named
# "name" and "id" where kv v1 is enabled at "secret/".
path "secret/profile" {
  capabilities = ["create"]
  required_parameters = ["name", "id"]
}
```

- `allowed_parameters` : A list of keys and values that are permitted on the given path.

  Setting a parameter with a value of the empty list allows the parameter to contain any value.

```
# This allows the user to update the password parameter value set on any
# users configured for userpass auth method. The password value can be
# anything. However, the user cannot update other parameter values such as
# "token_ttl".
path "auth/userpass/users/*" {
  capabilities = ["update"]
  allowed_parameters = {
    "password" = []
  }
}
```

  Setting a parameter with a value of a populated list allows the parameter to contain only those values.

```
# This allows the user to create or update an encryption key for transit
# secrets engine enabled at "transit/". When you do, you can set the
# "auto_rotate_period" parameter value so that the key gets rotated.
# However, the rotation period must be "8h", "24h", or "5d". Any other value
# will result in an error.
path "transit/keys/*" {
  capabilities = ["create", "update"]
  allowed_parameters = {
    "auto_rotate_period" = ["8h", "24h", "5d"]
  }
}
```

If any keys are specified, all non-specified parameters will be denied unless the parameter “*” is set to an empty array, which will allow all other parameters to be modified. Parameters with specific values will still be restricted to those values.

```
# When kv v1 secrets engine is enabled at "secret/", this allows the user to
# create "secret/foo" with a parameter named "bar". The parameter "bar" can
# only contain the values "zip" or "zap", but any other parameters may be
# created with any value.
path "secret/foo" {
  capabilities = ["create"]
  allowed_parameters = {
    "bar" = ["zip", "zap"]
    "*"   = []
  }
}
```

- `denied_parameters`: A list of keys and values that are not permitted on the given path. Any values specified here take precedence over `allowed_parameters`.

  Setting a parameter with a value of the empty list denies any changes to that parameter.

```
# This allows the user to update the userpass auth method's user
# configurations (e.g., "password") but cannot update the "token_policies"
# and "policies" parameter values.
path "auth/userpass/users/*" {
  capabilities = ["update"]
  denied_parameters = {
    "token_policies" = []
    "policies" = []
  }
}
```

  Setting a parameter with a value of a populated list denies any parameter containing those values.

```
# This allows the user to create or update token roles. However, the
# "allowed_policies" parameter value cannot be "admin", but the user can
# assign any other policies to the parameter.
path "auth/token/roles/*" {
  capabilities = ["create", "update"]
  denied_parameters = {
    "allowed_policies" = ["admin"]
  }
}
```

  Setting to `*` will deny any parameter.

```
# This allows the user to create or update an encryption key for transit
# secrets engine enabled at "transit/". However, the user cannot set any of
# the configuration parameters. As a result, the created key will have all
# parameters set to default values.
path "transit/keys/*" {
  capabilities = ["create", "update"]
  denied_parameters = {
    "*" = []
  }
}
```

If any parameters are specified, all non-specified parameters are allowed, unless `allowed_parameters` is also set, in which case normal rules apply.

Parameter values also support prefix/suffix globbing. Globbing is enabled by prepending or appending or prepending a splat (`*`) to the value:

```
# Only allow a parameter named "bar" with a value starting with "foo-*".
path "secret/foo" {
  capabilities = ["create"]
  allowed_parameters = {
    "bar" = ["foo-*"]
  }
}
```

The only value that can be used with the `*` parameter is `[]`.

Default values

Evaluation of policies with `allowed_parameters`, `denied_parameters`, and `required_parameters` happens without consideration of parameters' default values.

Given the following policy:

```
# The "no_store" parameter cannot be "false".
path "secret/foo" {
  capabilities = ["create"]
  denied_parameters = {
    "no_store" = [false, "false"]
  }
}
```

The following operation will error, because `no_store` is set to `false`:

```
d8 stronghold write secret/foo no_store=false value=bar
```

Whereas the following operation will succeed, even if the `no_store` parameter must be a boolean, and it defaults to `false`:

```
# Succeeds because "no_store=false" isn't present in the parameters.
d8 stronghold write secret/foo value=bar
```

This is because the policy evaluator does not know what the default value is for the `no_store` parameter. All it sees is that the denied parameter isn't present in the command.

This can be resolved by requiring the `no_store` parameter in your policy:

```
path "secret/foo" {
  capabilities = ["create"]
  required_parameters = ["no_store"]
  denied_parameters = {
    "no_store" = [false, "false"]
  }
}
```

The following command, which previously succeeded, will now fail under the new policy because there is no `no_store` parameter:

```
d8 stronghold write secret/foo value=bar
```

Globbing

It's also important to note that the use of globbing may result in surprising or unexpected behavior:

```
# This allows the user to create, update, or patch "secret/foo" with a parameter
# named "bar". the values passed to parameter "bar" must start with "baz/"
# so values like "baz/quux" are fine. however, values like
# "baz/quux,wibble,wobble,wubble" would also be accepted. the API that
# underlies "secret/foo" might allow comma delimited values for the "bar"
# parameter, and if it did, specifying a value like
# "baz/quux,wibble,wobble,wubble" would result in 4 different values getting
# passed along. seeing values like "wibble" or "wobble" getting passed to
# "secret/foo" might surprise someone that expected the allowed_parameters
# constraint to only allow values starting with "baz/".
path "secret/foo" {
  capabilities = ["create", "update", "patch"]
  allowed_parameters = {
    "bar" = ["baz/*"]
  }
}
```

Required response wrapping TTLs

These parameters can be used to set minimums and maximums on TTLs set by clients when requesting response wrapping, with a granularity of a second. These use `duration`-format strings.

In practice, setting a minimum TTL of one second effectively makes response wrapping mandatory for a particular path.

- `min_wrapping_ttl` : The minimum allowed TTL that clients can specify for a wrapped response. In practice, setting a minimum TTL of one second effectively makes response wrapping mandatory for a particular path. It can also be used to ensure that the TTL is not too low, leading to end targets being unable to unwrap before the token expires.
- `max_wrapping_ttl` : The maximum allowed TTL that clients can specify for a wrapped response.

```
# This effectively makes response wrapping mandatory for this path by setting "min_wra
pping_ttl" to 1 second.
# This also sets this path's wrapped response maximum allowed TTL to 90 seconds.
path "auth/approle/role/my-role/secret-id" {
    capabilities = ["create", "update"]
    min_wrapping_ttl = "1s"
    max_wrapping_ttl = "90s"
}
```

If both are specified, the minimum value must be less than the maximum. In addition, if paths are merged from different stanzas, the lowest value specified for each is the value that will result, in line with the idea of keeping token lifetimes as short as possible.

Built-in policies

Stronghold has two built-in policies: default and root. This section describes the two built-in policies.

Default policy

The default policy is a built-in Stronghold policy that cannot be removed. By default, it is attached to all tokens, but may be explicitly excluded at token creation time by supporting authentication methods.

The policy contains basic functionality such as the ability for the token to look up data about itself and to use its cubbyhole data. However, Stronghold is not prescriptive about its contents. It can be modified to suit your needs; Stronghold will never overwrite your modifications. If you want to stay up-to-date with the latest upstream version of the default policy, simply read the contents of the policy from an up-to-date dev server, and write those contents into your Stronghold's default policy.

To view all permissions granted by the default policy on your Stronghold installation, run:

```
d8 stronghold read sys/policy/default
```

To disable attachment of the default policy:

```
d8 stronghold token create -no-default-policy
```

or via the API:

```
curl \
  --request POST \
  --header "X-Vault-Token: ..." \
  --data '{"no_default_policy": "true"}' \
  https://stronghold.example.com/v1/auth/token/create
```

Root policy

The root policy is a built-in Stronghold policy that cannot be modified or removed. Any user associated with this policy becomes a root user. A root user can do anything within Stronghold. As such, it is highly recommended that you revoke any root tokens before running Stronghold in production.

When a Stronghold server is first initialized, there always exists one root user. This user is used to do the initial configuration and setup of Stronghold. After configured, the initial root token should be revoked and more strictly controlled users and authentication should be used.

To revoke a root token, run:

```
d8 stronghold token revoke "<token>"
```

or via the API:

```
curl \
  --request POST \
  --header "X-Vault-Token: ..." \
  --data '{"token": "<token>"}' \
  https://stronghold.example.com/v1/auth/token/revoke
```

Managing policies

Policies are authored (written) in your editor of choice. They can be authored in HCL or JSON, and the syntax is described in detail above. Once saved, policies must be uploaded to Stronghold before they can be used.

Listing policies

To list all registered policies in Stronghold:

```
d8 stronghold read sys/policy
```

or via the API:

```
curl \
  --header "X-Vault-Token: ..." \
  https://stronghold.example.com/v1/sys/policy
```

Creating policies

Policies may be created (uploaded) via the CLI or via the API. To create a new policy in Stronghold:

```
d8 stronghold policy write policy-name policy-file.hcl
```

or via the API:

```
curl \
  --request POST \
  --header "X-Vault-Token: ..." \
  --data '{"policy":"path \"...\" {...} "}' \
  https://stronghold.example.com/v1/sys/policy/policy-name
```

In both examples, the name of the policy is `policy-name`. You can think of this name as a pointer or symlink to the policy ACLs. Tokens are attached policies by name, which are then mapped to the set of rules corresponding to that name.

Updating policies

Existing policies may be updated to change permissions via the CLI or via the API. To update an existing policy in Stronghold, follow the same steps as creating a policy, but use an existing policy name:

```
d8 stronghold write sys/policy/my-existing-policy policy=@updated-policy.json
```

or via the API:

```
curl \
  --request POST \
  --header "X-Vault-Token: ..." \
  --data '{"policy":"path \"...\" {...} "}' \
  https://stronghold.example.com/v1/sys/policy/my-existing-policy
```

Deleting policies

Existing policies may be deleted via the CLI or API. To delete a policy:

```
d8 stronghold delete sys/policy/policy-name
```

or via the API:

```
curl \
  --request DELETE \
  --header "X-Vault-Token: ..." \
  https://stronghold.example.com/v1/sys/policy/policy-name
```

This is an idempotent operation. Stronghold will not return an error when deleting a policy that does not exist.

Associating policies

Stronghold can automatically associate a set of policies to a token based on an authorization. This configuration varies significantly between authentication backends. For simplicity, this example will use Stronghold's built-in Userpass auth method.

1. A Stronghold administrator or someone from the security team would create the user in Stronghold with a list of associated policies:

   ```
   d8 stronghold write auth/userpass/users/sethvargo \
       password="s3cr3t!" \
       policies="dev-readonly,logs"
   ```

2. This creates an authentication mapping to the policy such that, when the user authenticates successfully to Stronghold, they will be given a token which has the list of policies attached.

3. The user wishing to authenticate would run:

   ```
   d8 stronghold login -method="userpass" username="sethvargo"
   Password: ...
   ```

4. If the provided information is correct, Stronghold will generate a token, assign the list of configured policies to the token, and return that token to the authenticated user.

Root protected API endpoints

> Stronghold treats the HTTP POST and PUT verbs as equivalent, so for each mention of POST in the table above, PUT may also be used. Stronghold uses the non-standard LIST HTTP verb, but also allows list requests to be made using the GET verb along with `?list=true` as a query parameter, so for each mention of LIST in the table above, GET with `?list=true` may also be used.

The following paths requires a root token or `sudo` capability in the policy:

Path	HTTP verb	Description
`auth/token/accessors`	LIST	List token accessors for all current Stronghold service tokens
`auth/token/create`	POST	Create a periodic or an orphan token (`period` or `no_parent`) option
`pki/root`	DELETE	Delete the current CA key (pki secrets engine)
`pki/root/sign-self-issued`	POST	Use the configured CA certificate to sign a self-issued certificate (pki secrets engine)
`sys/audit`	GET	List enabled audit devices
`sys/audit/:path`	POST, DELETE	Enable or remove an audit device
`sys/auth/:path`	GET, POST, DELETE	Manage the auth methods (enable, read, and delete)
`sys/auth/:path/tune`	GET, POST	Manage the auth methods (enable, read, delete, and tune)
`sys/config/auditing/request-headers`	GET	List the request headers that are configured to be audited
`sys/config/auditing/request-headers/:name`	GET, POST, DELETE	Manage the auditing headers (create, update, read and delete)
`sys/config/cors`	GET, POST, DELETE	Configure CORS setting
`sys/config/ui/headers`	GET, LIST	Configure the UI settings
`sys/config/ui/headers/:name`	POST, DELETE	Configure custom HTTP headers to be served with the UI
`sys/internal/inspect/router/:tag`	GET	Inspect the internal components of Stronghold's router. `tag` must be one of `root` , `uuid` , `accessor` , or `storage`
`sys/leases/lookup/:prefix`	LIST	List lease IDs
`sys/leases/revoke-force/:prefix`	POST	Revoke all secrets or tokens ignoring backend errors
	POST	

Path	HTTP verb	Description
`sys/leases/revoke-prefix/:prefix`		Revoke all secrets generated under a given prefix
`sys/plugins/catalog/:type/:name`	GET, POST, DELETE	Register a new plugin, or read/remove an existing plugin
`sys/raw:path`	GET, POST, DELETE	Used to access the raw underlying store in Stronghold
`sys/raw:prefix`	GET, LIST	Returns a list keys for a given path prefix
`sys/remount`	POST	Moves an already-mounted backend to a new mount point
`sys/replication/reindex`	POST	Reindex the local data storage
`sys/replication/performance/primary/secondary-token`	POST	Generate a performance secondary activation token
`sys/replication/dr/primary/secondary-token`	POST	Generate a DR secondary activation token
`sys/rotate`	POST	Trigger a rotation of the backend encryption key
`sys/seal`	POST	Seals the Stronghold
`sys/step-down`	POST	Forces a node to give up active status
`sys/storage/raft/snapshot-auto/config`	LIST	Lists named configurations
`sys/storage/raft/snapshot-auto/config/:name`	GET, POST, DELETE	Creates or updates a named configuration

Tokens

Tokens have two sets of policies: identity policies, which are computed based on the entity and its groups, and token policies, which are either defined based on the login method or, in the case of explicit token creates via the API, are an input to the token creation. What follows concerns token policies exclusively: a token’s identity policies cannot be controlled except by modifying the underlying entities, groups, and group memberships.

Tokens are associated with their policies at creation time. For example:

```
d8 stronghold token create -policy=dev-readonly -policy=logs
```

Normally the only policies that may be specified are those which are present in the current token's (i.e. the new token's parent's) token policies. However, root users can assign any policies.

There is no way to modify the policies associated with a token once the token has been issued. The token must be revoked and a new one acquired to receive a new set of policies.

However, the contents of policies are parsed in real-time whenever the token is used. As a result, if a policy is modified, the modified rules will be in force the next time a token, with that policy attached, is used to make a call to Stronghold.

Password policy

The password policy (hereinafter referred to as the policy) defines a set of requirements for generated passwords. Stronghold allows you to configure custom policies and use them instead of the default policy.

Password policies are not related to access policies. They have similar names but serve different purposes.

Stronghold uses a password policy configuration model compatible with Vault `1.5+`.

Using password policies in Stronghold

In Stronghold, password policies are used for the following purposes:

- Generating passwords via the API that comply with a specific policy.
- Configuring automatic password generation using secret engines. Not all secret engines support password policies. Before configuring a policy for a secret engine, check its documentation.
- Validate a password created by a user using the `userpass` authentication method.

Policy structure and syntax

Password policies are defined in HCL or JSON.

The policy describes:

- The password length (specified in the `length` parameter).
- One or more rules that the password must satisfy. Each rule is described in the `rule` field. Stronghold allows the use of `charset` rules in policies.

> ⚠ A policy must contain at least one `charset` rule. A policy without a `charset` rule will be rejected.

Example of a policy:

```
length = 20

rule "charset" {
  charset = "abcdefghijklmnopqrstuvwxyz"
}
```

This policy generates a 20-character password using only lowercase Latin letters.

The length parameter

The `length` parameter sets the generated password length.

Characteristics:

- type: `int`;
- required;
- value must be at least `4`.

The charset rule

The `charset` rule defines a character set and, if needed, the minimum number of characters (`min-chars`) from this set that must be present in the password.

If several `charset` rules are specified in a policy, Stronghold combines all character sets and removes duplicates before generation starts (for example, if the sets `abcde` and `cdefg` are specified, the resulting set `abcdefg` will be used for generation). Each `charset` rule is still checked separately.

> ⚠ After combining and deduplicating character sets, the final `charset` used to generate candidate passwords must not contain more than `256` characters.

charset rule parameters

The following parameters are available in the `charset` rule:

Parameter name	Type	Default value	Description
`charset`	`string`	-	String representation of the character set. UTF-8 strings are supported. All characters must be printable
`min-chars`	`int`	`0`	Minimum number of characters from `charset` that must be present in the password

If `min-chars` is not specified or is set to `0`, the character set is used for generation but does not add a required constraint to the resulting password.

Example of using the `charset` rule in a policy:

```
length = 20

rule "charset" {
  charset = "abcde"
  min-chars = 1
}

rule "charset" {
  charset = "01234"
  min-chars = 1
}
```

This policy generates a password from the combined `abcde01234` character set. The password must contain at least one character from `abcde` and at least one character from `01234`.

Policy examples

Password 20 characters, only character sets, with no minimum requirement

```
length = 20

rule "charset" {
  charset = "abcdefghijklmnopqrstuvwxyz"
}

rule "charset" {
  charset = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
}

rule "charset" {
  charset = "0123456789"
}

rule "charset" {
  charset = "!\"#$%&'()*+,-./:;<=>?@[\\]^_`{|}~"
}
```

The password must be 20 characters long and include at least one uppercase letter, one lowercase letter, and one number

```
length = 20

rule "charset" {
  charset = "abcdefghijklmnopqrstuvwxyz"
  min-chars = 1
}

rule "charset" {
  charset = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
  min-chars = 1
}

rule "charset" {
  charset = "0123456789"
  min-chars = 1
}

rule "charset" {
  charset = "!\"#$%&'()*+,-./:;<=>?@[\\]^_`{|}~"
}
```

The password must be 20 characters long and include at least one uppercase letter, one lowercase letter, one digit, and one character from the full ASCII set of special characters

```
length = 20

rule "charset" {
  charset = "abcdefghijklmnopqrstuvwxyz"
  min-chars = 1
}

rule "charset" {
  charset = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
  min-chars = 1
}

rule "charset" {
  charset = "0123456789"
  min-chars = 1
}

rule "charset" {
  charset = "!\"#$%&'()*+,-./:;<=>?@[\\]^_`{|}~"
  min-chars = 1
}
```

The password must be 20 characters long and include at least one uppercase letter, one lowercase letter, one number, and one character from the set !@#$

```
length = 20

rule "charset" {
  charset = "abcdefghijklmnopqrstuvwxyz"
  min-chars = 1
}

rule "charset" {
  charset = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
  min-chars = 1
}

rule "charset" {
  charset = "0123456789"
  min-chars = 1
}

rule "charset" {
  charset = "!@#$"
  min-chars = 1
}
```

Default password policy

Stronghold uses the default password policy for passwords generated without an explicitly specified policy.

This policy requires:

- `20` password characters (for the `userpass` method — `8` characters);
- at least one uppercase letter;
- at least one lowercase letter;
- at least one digit;
- at least one dash character (`-`).

Default policy example:

```
length = 20

rule "charset" {
  charset = "abcdefghijklmnopqrstuvwxyz"
  min-chars = 1
}

rule "charset" {
  charset = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
  min-chars = 1
}

rule "charset" {
  charset = "0123456789"
  min-chars = 1
}

rule "charset" {
  charset = "-"
  min-chars = 1
}
```

Policy management

Creating and editing a policy

To create or edit a password policy, use the POST method on `/sys/policies/password/:name`.

Example of creating a policy from an HCL file:

```
d8 stronghold write sys/policies/password/my-policy policy=@my-policy.hcl
```

Example of passing a policy directly when creating it:

```
d8 stronghold write sys/policies/password/my-policy policy=- <<EOF
length = 20
rule “charset” {
  charset = “abcdefghijklmnopqrstuvwxyz0123456789”
}
EOF
```

To verify that the policy has been created, use the following command:

```
d8 stronghold read sys/policies/password/my-policy
```

Example of creating a policy via the API:

```
curl \
  --request POST \
  --header “X-Vault-Token: ...” \
  --data my-policy.json \
  https://stronghold.example.com/v1/sys/policies/password/my-policy
```

Getting a list of policies

To get a list of created policies, use the GET method on `/sys/policies/password`.

Example:

```
d8 stronghold read sys/policies/password
```

Example of getting a list of policies via the API:

```
curl \
  --header “X-Vault-Token: ...” \
  --request LIST \
  https://stronghold.example.com/v1/sys/policies/password
```

Retrieving policy information

To get information about a specific policy, use the GET method on `/sys/policies/password/:name`.

Example:

```
d8 stronghold read sys/policies/password/my-policy
```

Example of getting information about a specific policy via the API:

```
curl \
  --header “X-Vault-Token: ...” \
  https://stronghold.example.com/v1/sys/policies/password/my-policy
```

Deleting a policy

To delete a password policy, use the DELETE method on `/sys/policies/password/:name`.

Example:

```
d8 stronghold delete sys/policies/password/my-policy
```

Example of deleting a policy via the API:

```
curl \
  --header “X-Vault-Token: ...” \
  --request DELETE \
  https://stronghold.example.com/v1/sys/policies/password/my-policy
```

Generating passwords via the API

You can manually generate passwords that comply with a specific policy. To do this, use the `/sys/policies/password/:name/generate` method, replacing `:name` with the name of the policy the password must comply with.

Example:

```
d8 stronghold read sys/policies/password/my-policy/generate
```

Example of generating a password via the API:

```
curl \
  --header “X-Vault-Token: ...” \
  https://stronghold.example.com/v1/sys/policies/password/my-policy/generate
```

Principles and features of password generation

Stronghold first generates a candidate password from the combined character set built from all `charset` rules. Duplicate characters from different sets are removed. Stronghold then checks the candidate password against all rules.

Candidate password generation

The candidate password generation method affects security. The password is generated in such a way that the result cannot be predicted, and the generation process cannot be exploited to launch an attack.

The following are used to generate a candidate password:

1. A cryptographically secure random number generator.
2. A `charset` from which characters are selected.
3. The password length.

At a high level, the process looks like this:

1. Stronghold gets `N` random values, where `N` is the password length.
2. Each value is interpreted as an index in the `charset` array.
3. Characters are selected by these indexes.
4. The selected characters are combined into a candidate password.
5. The resulting password is checked against all policy rules.

For example, you need to generate an `8`-character password from the `abcdefghij` character set.

Stronghold gets the following random values:

```
[3, 2, 0, 8, 7, 3, 5, 1]
```

These values correspond to indexes in `charset`:

```
[3, 2, 0, 8, 7, 3, 5, 1] => [d, c, a, i, h, d, f, b]
```

The resulting candidate password is `dcaihdfb`.

Preventing bias

Restricting a random value to the size of the `charset` array with the modulo operation can introduce bias. This causes some characters to be selected more often than others.

For example, if the generator returns values in the `0-255` range and the `charset` length does not divide `256` evenly, the first characters in the set can appear more often.

Consider a simplified example. Suppose `charset` is `abcdefghij`, which contains `10` characters. Suppose the generator returns values in the `0-25` range.

In this case:

- values `0-9` correspond to all `10` characters;
- values `10-19` again correspond to all `10` characters;
- values `20-25` correspond only to the first `6` characters.

As a result, the `abcdef` characters are selected more often than `ghij`.

To avoid this behavior, Stronghold calculates the maximum allowed value that can be safely used to index `charset`. For this example, the maximum allowed value will be `19`. In this case, all values from `0` to `19`, inclusive, result in exactly 2 matches for each symbol. If a random number exceeds this value, it is discarded and generation continues. This keeps the final password length unchanged and preserves a correct character distribution.

Performance Characteristics

Password generation speed depends on the policy configuration. In general, stricter constraints require more time for generation.

A general estimate can be represented as:

```
password generation speed = (time to generate one candidate password) * (number of
generated candidate passwords)
```

The number of attempts depends on how likely it is that the next candidate password will fail validation against all rules.

In practice, this means the following:

- the more `min-chars` constraints there are, the higher the probability of regeneration;
- the smaller the required character subset is, the lower the probability of quickly getting a suitable password;
- as password length increases, the probability of satisfying some requirements increases, but the cost of generation also grows.

A noticeable performance drop occurs when one rule requires at least one character from a very small set, such as `!@#$`, while the overall `charset` is much larger.

To estimate performance for a specific policy, test it through the password generation API in your Stronghold environment.

Identity

This page contains information about *identity* along with an overview of the associated terms and concepts. The idea of identity is to maintain the clients accessing Stronghold. As such, Stronghold provides an identity management solution through the Identity secrets engine. For details about the Identity secrets engine, refer to the corresponding documentation section.

Terms

The terms “identity” and “entity” are related but represent different concepts.

Identity

Identity is a broader concept that involves managing the identification of users and applications. In Stronghold, identity enables centralized management of associated accounts and authentication methods. The identity system allows multiple authentication methods to be linked to a single user or application, providing flexibility and convenience when using different authentication mechanisms. Identity management is also closely tied to policy assignment and identity-based access control.

Entity

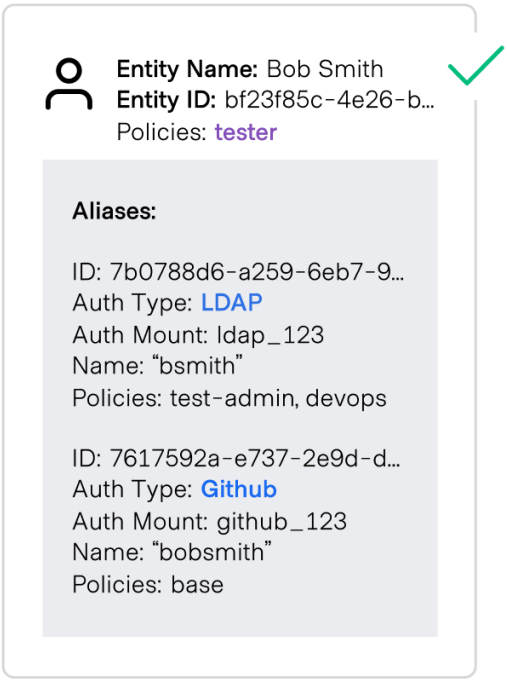
An entity is a specific implementation of the identity concept. It represents a single user or application that can use various authentication methods. Each entity can have multiple aliases that link it to different authentication methods. An entity consolidates all information about identity, authentication, and associated policies in one place, ensuring comprehensive access management.

Entities and aliases

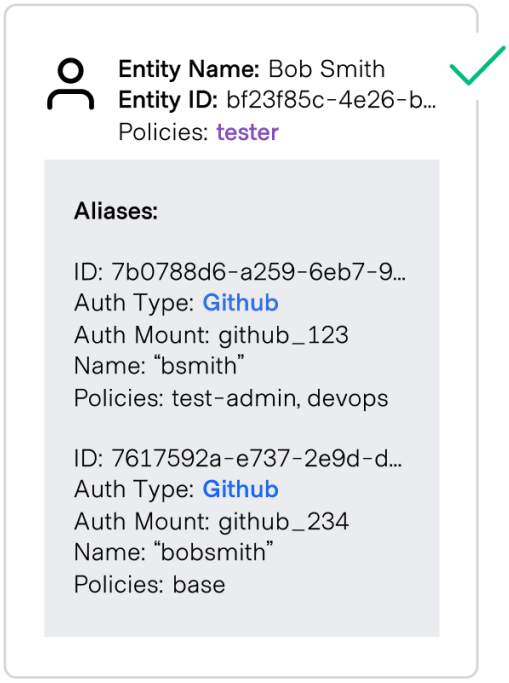
Each user may have multiple accounts with various identity providers, and Stronghold supports many of those providers to authenticate with Stronghold. Stronghold Identity can tie authentications from various

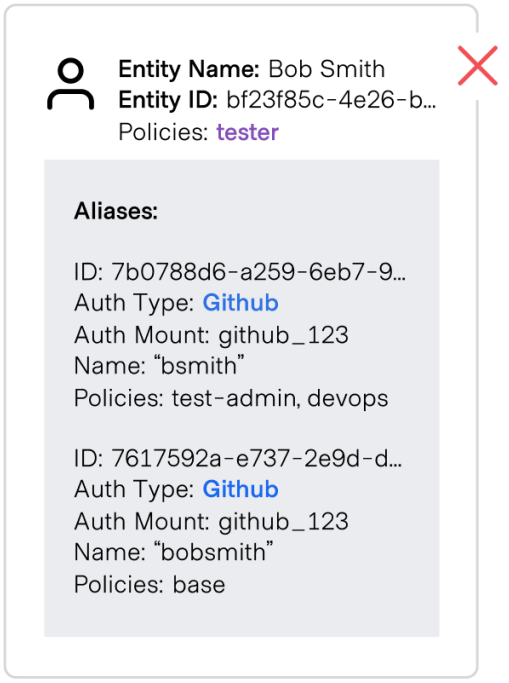
auth methods to a single representation. This representation of a consolidated identity is called an *entity* and their corresponding accounts with authentication providers can be mapped as *aliases*. In essence, each entity is made up of zero or more aliases. An entity cannot have more than one alias for a particular authentication backend.

For example, a user with accounts in both GitHub and LDAP can be mapped to a single entity in Stronghold with two aliases, one of type GitHub and one of type LDAP.



However, if both aliases are created on the same auth mount, such as a GitHub mount, both aliases cannot be mapped to the same entity. The aliases can have the same auth type, as long as the auth mounts are different, and still be associated to the same entity. The diagrams below illustrate both valid and invalid scenarios.





When a client authenticates via any credential backend (except the Token backend), Stronghold creates a new entity. It attaches a new alias to it if a corresponding entity does not already exist. The entity identifier will be tied to the authenticated token. When such tokens are used, their entity identifiers are audit logged, marking a trail of actions performed by specific users.

Entity management

Entities in Stronghold **do not** automatically pull identity information from anywhere. It needs to be explicitly managed by operators. This way, it is flexible in terms of administratively controlling the number of entities to be synced against Stronghold. In some sense, Stronghold will serve as a *cache* of identities and not as a *source* of identities.

Entity policies

Stronghold policies can be assigned to entities which will grant *additional* permissions to the token on top of the existing policies on the token. If the token presented on the API request contains an identifier for the entity and if that entity has a set of policies on it, then the token will be capable of performing actions allowed by the policies on the entity as well.

This is a paradigm shift in terms of *when* the policies of the token get evaluated. Before identity, the policy names on the token were immutable (not the contents of those policies though). But with entity policies, along with the immutable set of policy names on the token, the evaluation of policies applicable to the token through its identity will happen at request time. This also adds enormous flexibility to control the behavior of already issued tokens.

It is important to note that the policies on the entity are only a means to grant *additional* capabilities and not a replacement for the policies on the token. To know the full set of capabilities of the token with an associated entity identifier, the policies on the token should be taken into account.

> ⚠ Be careful in granting permissions to non-read-only identity endpoints. If a user can modify an entity, they can grant it additional privileges through policies. If a user can modify an alias they can login with, they can bind it to an entity with higher privileges. If a user can modify group membership, they can add their entity to a group with higher privileges.

Mount bound aliases

Stronghold supports multiple authentication backends and also allows enabling the same type of authentication backend on different mount paths. The alias name of the user will be unique within the backend's mount. But identity store needs to uniquely distinguish between conflicting alias names across different mounts of these identity providers. Hence, the alias name in combination with the authentication backend mount's accessor, serve as the unique identifier of an alias.

The table below shows what information each of the supported auth methods uses to form the alias name. This is the identifying information that is used to match or create an entity. If no entities are explicitly created or merged, then one entity will be implicitly created for each object on the right-hand side of the table, when it is used to authenticate on a particular auth mount point.

Auth method	Name reported by auth method
AppRole	Role ID
JWT/OIDC	Configurable via `user_claim` to one of the presented claims (no default value)
Kerberos	Username
Kubernetes	Configurable via `alias_name_source` to one of: Service account UID (default), Service account name
LDAP	Username
RADIUS	Username
TLS Certificate	Subject CommonName
Token	`entity_alias`, if provided
Username (userpass)	Username

Implicit entities

Operators can create entities for all the users of an auth mount beforehand and assign policies to them, so that when users login, the desired capabilities to the tokens via entities are already assigned. But if that's not done, upon a successful user login from any of the authentication backends, Stronghold will create a new entity and assign an alias against the login that was successful.

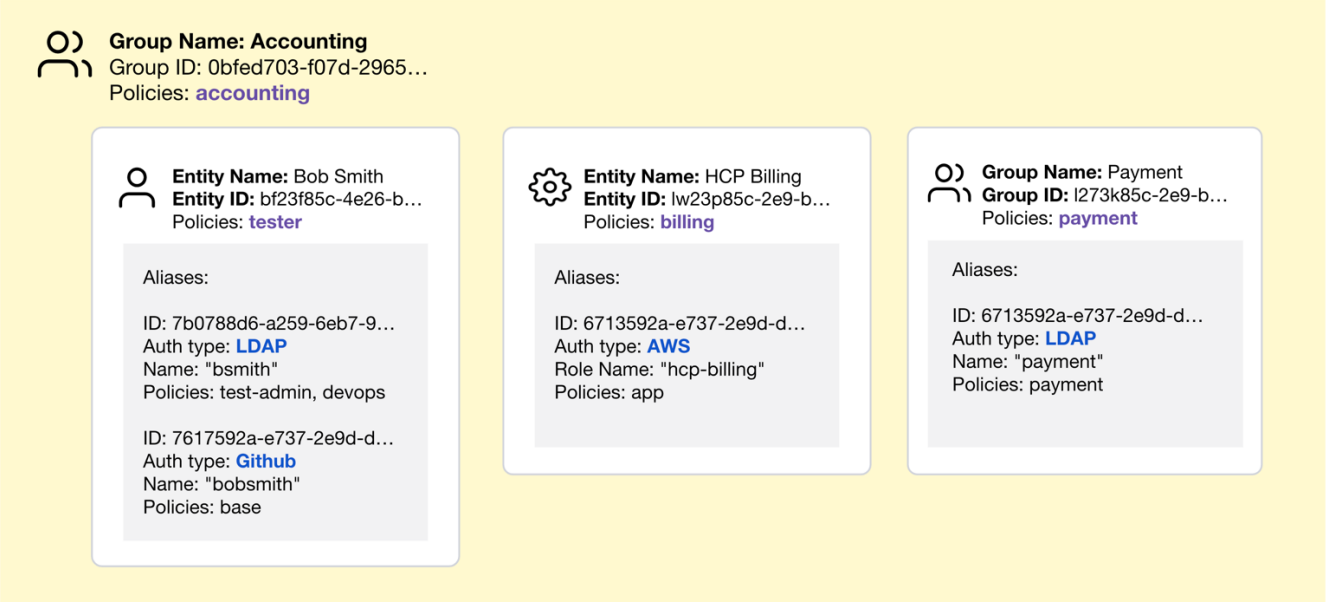
Note that the tokens created using the token authentication backend will not normally have any associated identity information. An existing or new implicit entity can be assigned by using the `entity_alias` parameter, when creating a token using a token role with a configured list of `allowed_entity_aliases`.

Identity auditing

If the token used to make API calls has an associated entity identifier, it will be audit logged as well. This leaves a trail of actions performed by specific users.

Identity groups

Stronghold Identity has support for *groups*. A group can contain multiple entities as its members. A group can also have subgroups. Policies set on the group are granted to all members of the group. During request time, when the token's entity ID is being evaluated for the policies that it has access to, policies that are inherited due to group memberships are granted along with the policies on the entity itself.



Group hierarchical permissions

Entities can be direct members of groups, in which case they inherit the policies of the groups they belong to. Entities can also be indirect members of groups. For example, if a Group A has Group B as subgroup, then members of Group B are indirect members of Group A. Hence, the members of Group B will have access to policies on both Group A and Group B.

External vs internal groups

By default, the groups created in identity store are called the internal groups. The membership management of these groups should be carried out manually. A group can also be created as an external group. In this case, the entity membership in the group is managed semi-automatically. An external group serves as a mapping to a group that is outside of the identity store. External groups can have one (and only one) alias. This alias should map to a notion of a group that is outside of the identity store.

For example, let's consider groups in LDAP and DEX. A username in LDAP belonging to a group in LDAP can get its entity ID added as a member of a group in Stronghold automatically during *logins* and *token renewals*. This works only if the group in Stronghold is an external group and has an alias that maps to the group in LDAP. If the user is removed from the group in LDAP, that change gets reflected in Stronghold only upon the subsequent login or renewal operation.

For details about the Identity secrets engine, refer to the corresponding documentation section.

Token

> ⚠ Tokens are opaque values so their structure is undocumented and subject to change. Scripts and automations that rely on the internal structure of a token in scripts will break.

Tokens can be used directly or auth methods can be used to dynamically generate tokens based on external identities.

All external authentication mechanisms map down to dynamically created tokens. These tokens have all the same properties as a normal manually created token.

Within Stronghold, tokens map to information. The most important information mapped to a token is a set of one or more attached policies. These policies control what the token holder is allowed to do within Stronghold. Other mapped information includes metadata that can be viewed and is added to the audit log, such as creation time, last renewal time, and more.

Token formats

Tokens are composed of a *prefix* and a *body*.

- The prefix indicates the token's type:

Token Type	Prefix
Service tokens	`s.`
Batch tokens	`b.`
Recovery tokens	`r.`

- The body is a randomly-generated string of 24 or more character (Base62 string).

Token are expected to match the following regexp: `[sbr]\.[a-zA-Z0-9]{24,}`

Examples:

```
b.n6keuKu5Q6pXhaIcfnC9cFNd
r.JaKnR2AIHNk3fC4SGyyyDVoQ9O
s.raPGTZdARXdY0KvHcWSpp5wWZIHNT
```

Token types

There are two types of tokens: `service` tokens and `batch` tokens. A section near the bottom of this page contains detailed information about their differences, but it is useful to understand other token concepts first. The features in the following sections all apply to service tokens, and their applicability to batch tokens is discussed later.

Token store

Often in documentation or in help channels, the "token store" is referenced. This is the same as the Token authentication backend. This is a special backend in that it is responsible for creating and storing

tokens, and cannot be disabled. It is also the only auth method that has no login capability – all actions require existing authenticated tokens.

Root tokens

Root tokens are tokens that have the root policy attached to them. Root tokens can do anything in Stronghold. In addition, they are the only type of token within Stronghold that can be set to never expire without any renewal needed. As a result, it is purposefully hard to create root tokens; in fact there are only three ways to create root tokens:

- The initial root token generated at `d8 stronghold operator init` time. This token has no expiration date.
- By using another root token. A root token with an expiration date cannot create a root token that never expires.
- By using `d8 stronghold operator generate-root` with the permission of a quorum of unseal key holders.

Token hierarchies and orphan tokens

Normally, when a token holder creates new tokens, these tokens will be created as children of the original token; tokens they create will be children of them; and so on. When a parent token is revoked, all of its child tokens – and all of their leases – are revoked as well. This ensures that a user cannot escape revocation by simply generating a never-ending tree of child tokens.

Often this behavior is not desired, so users with appropriate access can create orphan tokens. These tokens have no parent – they are the root of their own token tree. These orphan tokens can be created:

- Via `write` access to the `auth/token/create-orphan` endpoint.
- By having `sudo` or `root` access to the `auth/token/create` and setting the `no_parent` parameter to `true`.
- Via token store roles.
- By logging in with any other auth non-Token method.

Users with appropriate permissions can also use the `auth/token/revoke-orphan` endpoint, which revokes the given token but rather than revoke the rest of the tree, it instead sets the tokens' immediate children to be orphans. Use with caution!

Token accessors

When tokens are created, a token accessor is also created and returned. This accessor is a value that acts as a reference to a token and can only be used to perform limited actions:

- Look up a token's properties (not including the actual token ID)
- Look up a token's capabilities on a path
- Renew the token
- Revoke the token

Audit devices can optionally be set to not obfuscate token accessors in audit logs. This provides a way to quickly revoke tokens in case of an emergency. However, it also means that the audit logs can be used to perform a larger-scale DDoS attack.

Finally, the only way to list tokens is via the `auth/token/accessors` command, which actually gives a list of token accessors. While this is still a dangerous endpoint (since listing all of the accessors means that they can then be used to revoke all tokens), it also provides a way to audit and revoke the currently-active set of tokens.

Token Time-To-Live, periodic tokens, and explicit max TTLs

Every non-root token has a time-to-live (TTL) associated with it, which is a current period of validity since either the token's creation time or last renewal time, whichever is more recent. Root tokens may have a TTL associated, but the TTL may also be 0, indicating a token that never expires. After the current TTL is up, the token will no longer function – it, and its associated leases, are revoked.

If the token is renewable, Stronghold can be asked to extend the token validity period using `d8 stronghold token renew` or the appropriate renewal endpoint. At this time, various factors come into play. What happens depends upon whether the token is a periodic token (available for creation by `root` / `sudo` users, token store roles, or some auth methods), has an explicit maximum TTL attached, or neither.

General case

In the general case, where there is neither a period nor explicit maximum TTL value set on the token, the token's lifetime since it was created will be compared to the maximum TTL. This maximum TTL value is dynamically generated and can change from renewal to renewal, so the value cannot be displayed when a token's information is looked up. It is based on a combination of factors:

1. The system max TTL, which is 32 days but can be changed.
2. The max TTL set on a mount using `mount tuning`. This value is allowed to override the system max TTL – it can be longer or shorter, and if set this value will be respected.
3. A value suggested by the auth method that issued the token. This might be configured on a per-role, per-group, or per-user basis. This value is allowed to be less than the mount max TTL (or, if not set, the system max TTL), but it is not allowed to be longer.

Note that the values in (2) and (3) may change at any given time, which is why a final determination about the current allowed max TTL is made at renewal time using the current values. It is also why it is important to always ensure that the TTL returned from a renewal operation is within an allowed range; if this value is not extending, likely the TTL of the token cannot be extended past its current value and the client may want to reauthenticate and acquire a new token. However, outside of direct operator interaction, Stronghold will never revoke a token before the returned TTL has expired.

Explicit max TTLs

Tokens can have an explicit max TTL set on them. This value becomes a hard limit on the token's lifetime — no matter what the values in (1), (2), and (3) from the general case are, the token cannot live past this explicitly-set value. This has an effect even when using periodic tokens to escape the normal TTL mechanism.

Periodic tokens

In some cases, having a token be revoked would be problematic — for instance, if a long-running service needs to maintain its SQL connection pool over a long period of time. In this scenario, a periodic token can be used. Periodic tokens can be created in a few ways:

- By having `sudo` capability or a `root` token with the `auth/token/create` endpoint.
- By using token store roles.
- By using an auth method that supports issuing these, such as AppRole.

At issue time, the TTL of a periodic token will be equal to the configured period. At every renewal time, the TTL will be reset back to this configured period, and as long as the token is successfully renewed within each of these periods of time, it will never expire. Outside of `root` tokens, it is currently the only way for a token in Stronghold to have an unlimited lifetime.

The idea behind periodic tokens is that it is easy for systems and services to perform an action relatively frequently — for instance, every two hours, or even every five minutes. Therefore, as long as a system is actively renewing this token — in other words, as long as the system is alive — the system is allowed to keep using the token and any associated leases. However, if the system stops renewing within this period (for instance, if it was shut down), the token will expire relatively quickly. It is good practice to keep this period as short as possible, and generally speaking it is not useful for humans to be given periodic tokens.

There are a few important things to know when using periodic tokens:

- When a periodic token is created via a token store role, the *current* value of the role's period setting will be used at renewal time.
- A token with both a period and an explicit max TTL will act like a periodic token but will be revoked when the explicit max TTL is reached.

CIDR-bound tokens

Some tokens are able to be bound to CIDR(s) that restrict the range of client IPs allowed to use them. These affect all tokens except for non-expiring root tokens (those with a TTL of zero). If a root token has an expiration, it also is affected by CIDR-binding.

Token types in detail

There are currently two types of tokens.

Service tokens

Service tokens are what users will generally think of as “normal” Stronghold tokens. They support all features, such as renewal, revocation, creating child tokens, and more. They are correspondingly heavyweight to create and track.

Batch tokens

Batch tokens are encrypted blobs that carry enough information for them to be used for Stronghold actions, but they require no storage on disk to track them. As a result they are extremely lightweight and scalable, but lack most of the flexibility and features of service tokens.

Token type comparison

This reference chart describes the difference in behavior between service and batch tokens.

	Service Tokens	Batch Tokens
Can Be Root Tokens	Yes	No
Can Create Child Tokens	Yes	No
Can be Renewable	Yes	No
Manually Revocable	Yes	No
Can be Periodic	Yes	No
Can have Explicit Max TTL	Yes	No (always uses a fixed TTL)
Has Accessors	Yes	No
Has Cubbyhole	Yes	No
Revoked with Parent (if not orphan)	Yes	Stops Working
Dynamic Secrets Lease Assignment	Self	Parent (if not orphan)
Cost	Heavyweight; multiple storage writes per token creation	Lightweight; no storage cost for token creation

Service vs. batch token lease handling

Service tokens

Leases created by service tokens (including child tokens’ leases) are tracked along with the service token and revoked when the token expires.

Batch tokens

Leases created by batch tokens are constrained to the remaining TTL of the batch tokens and, if the batch token is not an orphan, are tracked by the parent. They are revoked when the batch token’s TTL expires, or when the batch token’s parent is revoked (at which point the batch token is also denied access to Stronghold).

Install

Running on Linux OS

Installation

Stronghold supports multi-server mode to ensure high availability (HA). This mode is automatically enabled when using a compatible data storage backend and protects the system from failures by running multiple Stronghold servers.

To check if your storage supports high availability mode, start the server and make sure the message `HA available` is displayed next to the storage information. In this case, Stronghold will automatically use HA mode.

To provide high availability, one of the Stronghold nodes acquires a lock in the storage system and becomes active, while the other nodes switch to standby mode. If standby nodes receive requests, they either forward them or redirect clients according to the configuration and the current state of the cluster.

To run Stronghold in high availability (HA) mode with the integrated Raft storage, you need at least three Stronghold servers. This requirement is necessary to achieve quorum — without it, the cluster cannot operate with the storage.

Prerequisites:

- A supported OS is installed on the server (Ubuntu, RedOS, Astra Linux).
- The Stronghold distribution is copied to the server.
- A systemd unit is created to manage the service.
- Individual certificates are issued for each node in the Raft cluster.
- A root certificate authority (CA) certificate is prepared.

Infrastructure preparation

The following scenario describes the deployment of a Stronghold cluster consisting of three nodes: one active and two standby. Such a cluster provides high availability (HA).

Launch via systemd unit

> ⚠ All examples assume that a system user `stronghold` has been created and the service runs under this user. If you need to use another user, replace `stronghold` with the appropriate name.

1. Create the file `/etc/systemd/system/stronghold.service` with the following content:

```
[Unit]
Description=Stronghold service
Documentation=https://deckhouse.io/products/stronghold/
After=network.target

[Service]
Type=simple
ExecStart=/opt/stronghold/stronghold server -config=/opt/stronghold/config.hcl
ExecReload=/bin/kill -HUP $MAINPID
KillMode=process
Restart=on-failure
RestartSec=5
User=stronghold
Group=stronghold
LimitNOFILE=65536
CapabilityBoundingSet=CAP_IPC_LOCK
AmbientCapabilities=CAP_IPC_LOCK
SecureBits=noroot

[Install]
WantedBy=multi-user.target
```

2. Apply the systemd configuration changes:

```
systemctl daemon-reload
```

3. Enable the service to start automatically:

```
systemctl enable stronghold.service
```

4. Create the `/opt/stronghold/data` directory and set the appropriate permissions:

```
mkdir -p /opt/stronghold/data
chown stronghold:stronghold /opt/stronghold/data
chmod 0700 /opt/stronghold/data
```

Preparing the required certificates

To configure TLS, you need a set of certificates and keys that must be placed in the `/opt/stronghold/tls` directory:

- Root certificate authority (CA).
  `stronghold-ca.pem` — the certificate used to sign Stronghold TLS certificates.
- Raft node certificates. In this scenario, the cluster will include three nodes, for which the following certificates will be created:
  - `node-1-cert.pem`;

    - `node-2-cert.pem`;
    - `node-3-cert.pem`.
- Private keys of the node certificates:
    - `node-1-key.pem`;
    - `node-2-key.pem`;
    - `node-3-key.pem`.

In this example, a root certificate and a set of self-signed certificates for each node will be created.

> ⚠ Self-signed certificates are suitable only for testing and experimentation.
> For production use, it is strongly recommended to use certificates issued and signed by a trusted certificate authority (CA).

Procedure

1. On the first node, create a directory for storing certificates (if it does not already exist) and switch to it:

   ```
   mkdir -p /opt/stronghold/tls
   cd /opt/stronghold/tls/
   ```

2. Generate a key for the root certificate:

   openssl genrsa 2048 > stronghold-ca-key.pem

3. Issue the root certificate:

   ```
   openssl req -new -x509 -nodes -days 3650 -key stronghold-ca-key.pem -out
   stronghold-ca.pem

   Country Name (2 letter code) [XX]:RU
   Locality Name (eg, city) [Default City]:Moscow
   Organization Name (eg, company) [Default Company Ltd]:MyOrg
   Common Name (eg, your name or your server hostname) []:demo.tld
   ```

   > The certificate attributes are provided as an example.

4. To issue node certificates, create configuration files that contain the `subjectAltName` (SAN) parameter.
   For example, for the `raft-node-1` node:

```
cat << EOF > node-1.cnf
[v3_ca]
subjectAltName = @alt_names
[alt_names]
DNS.1 = raft-node-1.demo.tld
IP.1 = 10.20.30.10
IP.2 = 127.0.0.1
EOF
```

Each node must have valid FQDN and IP addresses.
The `subjectAltName` field in the certificate must contain values relevant to the specific node.
Similarly, create a separate configuration file for each node.

5. Generate certificate signing requests (CSRs) and keys for the nodes:

```
openssl req -newkey rsa:2048 -nodes -keyout node-1-key.pem -out node-1-csr.pem -
subj "/CN=raft-node-1.demo.tld"
openssl req -newkey rsa:2048 -nodes -keyout node-2-key.pem -out node-2-csr.pem -
subj "/CN=raft-node-2.demo.tld"
openssl req -newkey rsa:2048 -nodes -keyout node-3-key.pem -out node-3-csr.pem -
subj "/CN=raft-node-3.demo.tld"
```

6. Issue certificates based on the created CSRs:

```
openssl x509 -req -set_serial 01 -days 3650 -in node-1-csr.pem -out node-1-
cert.pem -CA stronghold-ca.pem -CAkey stronghold-ca-key.pem -extensions v3_ca -
extfile ./node-1.cnf
openssl x509 -req -set_serial 02 -days 3650 -in node-2-csr.pem -out node-2-
cert.pem -CA stronghold-ca.pem -CAkey stronghold-ca-key.pem -extensions v3_ca -
extfile ./node-2.cnf
openssl x509 -req -set_serial 03 -days 3650 -in node-3-csr.pem -out node-3-
cert.pem -CA stronghold-ca.pem -CAkey stronghold-ca-key.pem -extensions v3_ca -
extfile ./node-3.cnf
```

> It is recommended to use unique `-set_serial` values for each certificate.

7. Copy the required files to each node:
    - Node certificate;
    - Node private key;
    - Root certificate.

   For example, for the `raft-node-2` and `raft-node-3` nodes:

```
scp ./node-2-key.pem ./node-2-cert.pem ./stronghold-ca.pem raft-
node-2.demo.tld:/opt/stronghold/tls
scp ./node-3-key.pem ./node-3-cert.pem ./stronghold-ca.pem raft-
node-3.demo.tld:/opt/stronghold/tls
```

> If the `/opt/stronghold/tls` directory does not exist on the target nodes, create it.

Deploying a Raft cluster

1. Connect to the first server where the Stronghold cluster initialization will be performed.
2. Allow network connections for TCP ports 8200 and 8201. Example for firewalld:

   ```
   firewall-cmd --add-port=8200/tcp --permanent
   firewall-cmd --add-port=8201/tcp --permanent
   firewall-cmd --reload
   ```

   > If necessary, you can use other ports by specifying them in the `/opt/stronghold/config.hcl` configuration file.

3. Create the /opt/stronghold/config.hcl configuration file for Raft. If the `/etc/stronghold/` directory does not exist, create it:

```
ui = true
cluster_addr  = "https://10.20.30.10:8201"
api_addr      = "https://10.20.30.10:8200"
disable_mlock = true

listener "tcp" {
  address            = "0.0.0.0:8200"
  tls_cert_file      = "/opt/stronghold/tls/node-1-cert.pem"
  tls_key_file       = "/opt/stronghold/tls/node-1-key.pem"
}

storage "raft" {
  path    = "/opt/stronghold/data"
  node_id = "raft-node-1"

  retry_join {
    leader_tls_servername   = "raft-node-1.demo.tld"
    leader_api_addr         = "https://10.20.30.10:8200"
    leader_ca_cert_file     = "/opt/stronghold/tls/stronghold-ca.pem"
    leader_client_cert_file = "/opt/stronghold/tls/node-1-cert.pem"
    leader_client_key_file  = "/opt/stronghold/tls/node-1-key.pem"
  }
  retry_join {
    leader_tls_servername   = "raft-node-2.demo.tld"
    leader_api_addr         = "https://10.20.30.11:8200"
    leader_ca_cert_file     = "/opt/stronghold/tls/stronghold-ca.pem"
    leader_client_cert_file = "/opt/stronghold/tls/node-1-cert.pem"
    leader_client_key_file  = "/opt/stronghold/tls/node-1-key.pem"
  }
  retry_join {
    leader_tls_servername   = "raft-node-3.demo.tld"
    leader_api_addr         = "https://10.20.30.12:8200"
    leader_ca_cert_file     = "/opt/stronghold/tls/stronghold-ca.pem"
    leader_client_cert_file = "/opt/stronghold/tls/node-1-cert.pem"
    leader_client_key_file  = "/opt/stronghold/tls/node-1-key.pem"
  }
}
```

4. Start the Stronghold service:

```
systemctl start stronghold
```

5. Initialize the cluster:

```
stronghold operator init -ca-cert /opt/stronghold/tls/stronghold-ca.pem
```

If necessary, you can specify the following parameters:

- `-key-shares` — number of key shares (default: 5);
- `-key-threshold` — minimum number of shares required to unseal the storage (default: 3).

> After initialization, all key shares and the root token will be displayed in the terminal. Be sure to save them in a secure place. Without the required number of key shares, access to Stronghold data will be impossible.

6. Unseal the cluster. Run the command multiple times, entering the unseal keys:

```
stronghold operator unseal -ca-cert /opt/stronghold/tls/stronghold-ca.pem
```

> If the `-key-threshold` parameter was not changed, you need to enter 3 key shares.

7. Configure the remaining nodes:
   - Set the appropriate cluster_addr and api_addr values in `/opt/stronghold/config.hcl`.
   - Skip the initialization step.
   - Immediately proceed to unsealing the cluster (operator unseal).
8. Verify the cluster status:

```
stronghold status -ca-cert /opt/stronghold/tls/stronghold-ca.pem
Key                     Value
---                     -----
Seal Type               shamir
Initialized             true
Sealed                  false
Total Shares            5
Threshold               3
Version                 1.15.2
Build Date              2025-03-07T16:10:46Z
Storage Type            raft
Cluster Name            stronghold-cluster-a3fcc270
Cluster ID              f682968d-5e6c-9ad4-8303-5aecb259ca0b
HA Enabled              true
HA Cluster              https://10.20.30.10:8201
HA Mode                 active
Active Node Address     https://10.20.30.10:8200
Raft Committed Index    40
Raft Applied Index      40
```

Configuration

Configuration overview

Stronghold servers are configured using a configuration file in either HCL or JSON format.

To enhance control over file access, you can enable file permission checks by setting the `VAULT_ENABLE_FILE_PERMISSIONS_CHECK` environment variable.
When this check is enabled, Stronghold verifies that the configuration directory and files are owned by the user running Stronghold.
It also ensures that neither the group nor other users have write or execute permissions on these files.

If necessary, the operator can specify the user and file permissions for the plugin directory and executable files.
This is done using the `plugin_file_uid` and `plugin_file_permissions` parameters in the configuration.
By default, file permission checks in Stronghold are disabled.

Here is an example configuration:

```
ui            = true
cluster_addr  = "https://127.0.0.1:8201"
api_addr      = "https://127.0.0.1:8200"
disable_mlock = true

storage "raft" {
  path = "/path/to/raft/data"
  node_id = "raft_node_id"
}

listener "tcp" {
  address       = "127.0.0.1:8200"
  tls_cert_file = "/path/to/full-chain.pem"
  tls_key_file  = "/path/to/private-key.pem"
}

telemetry {
  statsite_address = "127.0.0.1:8125"
  disable_hostname = true
}
```

To apply new parameters after modifying the configuration file, you need to restart the Stronghold service:
`systemctl restart stronghold`.

Parameter overview

- `storage` — **required** block. Configures the storage backend where Stronghold data will be stored.
  To run Stronghold in High Availability (HA) mode, you must use a backend that supports coordination semantics.
  If the selected storage backend supports this, HA parameters can be specified directly within the `storage` block.
  Otherwise, you should configure a separate `ha_storage` parameter with a backend that supports HA, along with the corresponding HA parameters.
  Details on the storage backend parameters are in the storage section.

- `ha_storage` — optional block. Configures the storage backend where Stronghold coordination in High Availability (HA) mode will occur.
  The specified backend must support HA. If this parameter is not set, Stronghold will attempt to run HA on the backend specified in the `storage` parameter.
  If the storage backend already supports HA coordination and the specific HA parameters are already specified in the `storage` block, additional `ha_storage` configuration is not required.

- `listener` — **required** block. Configures the parameters for listening to Stronghold API requests.
  Details are in the listener section.

- `user_lockout` — optional block. Configures the behavior for user lockout after failed login attempts.
  Details are in the user_lockout section.

- `cluster_name` — optional string. Specifies the identifier for the Stronghold cluster. If no value is provided, Stronghold will generate one.

- `cache_size` — optional string. Defines the size of the read cache used by the physical storage subsystem.
  The value is specified as the number of stored records, so the total cache size depends on record size. Default is `131072`.

- `disable_cache` — optional boolean. Disables all caches in Stronghold, including the read cache used by the physical storage subsystem.
  This significantly impacts performance. Default is `false`.

- `disable_mlock` — optional boolean. Disables the server's ability to perform the `mlock` system call, which prevents memory from being paged to disk.

  It is **not recommended** to disable `mlock` unless using the integrated storage.
  When disabling `mlock`, you should follow the additional security measures described below.

  It is **not recommended** to disable `mlock` if the systems running Stronghold either do not use swap or use only encrypted swap.
  Memory lock support is available only on UNIX-like systems such as Linux or FreeBSD that

support the `mlock()` system call.
Systems like Windows, NaCL, and Android do not have mechanisms to prevent the entire process address space from being written to disk, so this feature is automatically disabled for unsupported platforms.

On the contrary, it is **strongly recommended** to disable `mlock` when using integrated storage. This system call works poorly with memory-mapped files, such as those created by BoltDB, which Raft uses for state tracking.

Using `mlock` with integrated storage can cause memory shortages if Stronghold's data volume exceeds available RAM.
Memory-mapped files are loaded into resident memory, causing all Stronghold data to be loaded into RAM.
In this case, even though BoltDB data remains encrypted at rest, swap should be disabled to prevent other sensitive Stronghold data in memory from being paged to disk.

On Linux, you can allow the Stronghold executable to use the `mlock` system call without running the process as root by executing the following command:

```
sudo setcap cap_ipc_lock=+ep $(readlink -f $(which stronghold))
```

Each plugin runs as a separate process, so you need to apply similar settings for each plugin in the `plugins` directory.
If you are using a Linux distribution with a current version of `systemd`, you can add the following directive to the `[Service]` section of the configuration file:

```
LimitMEMLOCK=infinity
```

- `disable_sealwrap` — disables using seal wrapping for any value except the root key. If this value is toggled, the new behavior will happen lazily (as values are read or written).
- `plugin_directory` — optional string parameter. Defines the directory from which plugins are allowed to be loaded. For successful plugin loading, Stronghold must have read permissions for the files in this directory, and the value cannot be a symbolic link. By default, this parameter is not set and has the value `""`.
- `plugin_tmpdir` — optional string parameter. Specifies the directory where Stronghold can create temporary files to support interaction with Unix sockets for containerized plugins. If this value is not set, Stronghold will use the default temporary file directory. In most cases, this parameter does not need to be configured, except when containerized plugins are used and Stronghold shares the temporary folder with other processes, such as when using the `PrivateTmp` parameter in systemd.
- `plugin_file_uid` — optional integer parameter. User ID (UID) for plugin directories and executable files in case they belong to a user different from the one running Stronghold.

- `plugin_file_permissions` — optional string parameter. A string of octal permissions for plugin directories and executable files in case write or execute permissions are set for the group or other users.
- `telemetry` — optional block of parameters. Specifies the telemetry system for collecting and sending statistics.
- `default_lease_ttl` — optional string parameter. Defines the default lease TTL for tokens and secrets. The value is specified using a time suffix, such as `"40s"` (40 seconds) or `"1h"` (1 hour), and cannot exceed `max_lease_ttl`. By default, `768h`.
- `max_lease_ttl` — optional string parameter. Defines the maximum lease TTL for tokens and secrets. The value is specified using a time suffix, such as `"40s"` (40 seconds) or `"1h"` (1 hour). Specific mount points can override this value by configuring the mount point using the `max-lease-ttl` flag in the `auth` or `secret` commands. By default, `768h`.
- `default_max_request_duration` — optional string parameter. Specifies the maximum default request duration after which Stronghold will cancel it. The value is specified using a time suffix, such as `"40s"` (40 seconds) or `"1h"` (1 hour), and can be overridden for each listener through the `max_request_duration` parameter. By default, `90s`.
- `detect_deadlocks` — optional string parameter. A comma-separated string of values. Specifies internal mutually exclusive locks to watch for potential deadlocks. Currently supported values are `statelock`, `quotas`, and `expiration`, which will log `"POTENTIAL DEADLOCK:"` if a kernel state lock attempt seems to be blocked. Enabling `detect_deadlocks` may negatively impact performance due to tracking each lock attempt. By default, this parameter is not set and has the value `""`.
- `raw_storage_endpoint` — optional boolean parameter. Activates the `sys/raw` endpoint with elevated privileges. This endpoint allows decryption and encryption of raw data coming in and out of the security barrier. By default, `false`.
- `introspection_endpoint` — optional boolean parameter. Activates the `sys/internal/inspect` endpoint, which allows inspection of specific subsystems within Stronghold by users with root token or sudo privileges. By default, `false`.
- `ui` — optional boolean parameter. Activates the built-in web user interface, available on all listeners (address + port) at the path `/ui`. When browsers access the default Stronghold API address, they will automatically be redirected to the UI. By default, `false`. Details are in the ui section.
- `pid_file` — optional string parameter. Specifies the path to the file where the Stronghold process ID (PID) should be stored.
- `enable_response_header_hostname` — optional boolean parameter. Activates the addition of the `X-Vault-Hostname` HTTP header to all Stronghold HTTP responses. This header will contain the hostname of the Stronghold node that handled the HTTP request. The presence of this information

is not guaranteed — it will be provided if possible. If the option is enabled but the `X-Vault-Hostname` header is missing from the response, it may indicate an error extracting the hostname from the operating system. By default, `false`.

- `enable_response_header_raft_node_id` — optional boolean parameter. Activates the addition of the `X-Vault-Raft-Node-ID` HTTP header to all Stronghold HTTP responses. If Stronghold uses integrated storage (i.e., is part of a Raft cluster), the `X-Vault-Raft-Node-ID` header will contain the Raft node ID that handled the HTTP request. If the Stronghold node is not part of a Raft cluster, this header will be omitted regardless of whether the option is enabled. By default, `false`.
- `log_level` — optional string parameter. Defines the level of log detail. The following five values are supported in decreasing order of detail: `trace`, `debug`, `info`, `warn`, and `error`. This value can also be set via the environment variable `VAULT_LOG_LEVEL`. By default, `info`.

  If a valid `log_level` value is provided, when SIGHUP (`sudo kill -s HUP pid Stronghold`) is sent, Stronghold will update the existing log level. Both the CLI flag and the environment variable will be overridden. Not all parts of the Stronghold log may dynamically update. For example, secret/auth plugins are currently not dynamically updated.
- `log_format` — optional string parameter. The log format. Two values are supported: `standard` and `json`. By default, `standard`.
- `log_file` — optional string parameter. The absolute path where Stronghold should save log messages in addition to other existing outputs such as journald/stdout. Paths that end with a path separator use the default file name — `vault.log`. Paths that don't end with an extension use the default extension — `.log`. If the log file is overwritten, Stronghold appends the current timestamp to the file name at the time of overwriting.
- `log_rotate_duration` — optional string parameter. Specifies the maximum duration of writing to the log file before it is overwritten. The value is specified using a time suffix, such as `40s`. By default, `24h`.
- `log_rotate_bytes` — optional integer parameter. Specifies the number of bytes that can be written to the log file before it is overwritten. If not specified, the number of bytes written to the log file is unlimited.
- `log_rotate_max_files` — optional integer parameter. Specifies the maximum number of old log files to keep. By default, the value is `0`, meaning files are never deleted. To delete old log files when creating a new one, set the value to `-1`.
- `imprecise_lease_role_tracking` — optional boolean parameter. Allows skipping the lease count by roles if role-based quotas are not enabled. When set to `true` and new role-based quotas are enabled, the subsequent lease count will start at 0. This parameter affects role-based lease quotas but reduces latency when role quotas are not used.
- `experiments` — optional array of values. A list of experimental features to activate for the node. Do not use experimental features in production environments! Associated APIs may undergo

incompatible changes between releases. Additional experimental features can also be specified through the environment variable `VAULT_EXPERIMENTS` as a comma-separated list of values.

High availability parameters

For backends that support High Availability (HA) mode, the following parameters are used:

- `api_addr` — optional string parameter. Specifies the address that will be advertised to other Stronghold servers in the cluster for client redirection. This value is also used for plugin backends. The `api_addr` parameter can also be set through the environment variable `VAULT_API_ADDR`. Generally, this should be a full URL pointing to the listener's address. The address can be dynamically determined using the go-sockaddr template, which is resolved at runtime.
- `cluster_addr` — optional string parameter. Specifies the address to be advertised to other Stronghold servers in the cluster for request redirection. This parameter can also be set through the environment variable `VAULT_CLUSTER_ADDR`. Similar to `api_addr`, this is a full URL, but Stronghold will ignore the scheme (all cluster members always use TLS with a private key/certificate). The address can be dynamically determined using the go-sockaddr template, which is resolved at runtime.
- `disable_clustering` — optional boolean parameter. Specifies whether clustering features, such as request forwarding, are enabled. Setting the value to `true` for one storage node will disable clustering features only if that node is active. The parameter cannot be set to `true` if the storage type is `raft`. By default, `false`.

Listener section

The `listener` section configures the addresses and ports where Stronghold will listen for requests. Currently, there are two types of listeners:

1. TCP.
2. Unix Domain Socket.

TCP

The TCP listener configures Stronghold to listen on a TCP address/port.

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

You can specify the `listener` section multiple times to have Stronghold listen on multiple interfaces. When configuring multiple listeners, be sure to set the `api_addr` and `cluster_addr` parameters so that Stronghold advertises the correct address to other nodes.

Hiding Confidential Data for Unauthenticated Endpoints

Unauthenticated API endpoints may return the following confidential information:

1. The Stronghold version number.
2. The build date of the Stronghold binary.
3. The name of the Stronghold cluster.
4. The IP address of nodes in the cluster.

Stronghold allows configuring each `tcp` `listener` section to remove this data from API responses if necessary. The removal of sensitive information based on the listener section configuration is supported for the following three API endpoints:

1. `/sys/health`
2. `/sys/leader`
3. `/sys/seal-status`

The removed information is replaced by an empty string `""`. Additionally, some Stronghold API responses omit keys if the corresponding value is empty (`""`).

> ⚠ Removing values affects responses for all API clients. The Stronghold command-line interface (CLI) and user interface use Stronghold API responses. As a result, the removal settings will apply to the output in the CLI and user interface, as well as to direct API calls.

Custom Response Headers

Stronghold allows setting custom HTTP response headers for the root path (`/`) and for API endpoints (`/v1/`). These headers are determined based on the returned status code. For example, a user can define one set of custom response headers for the `200` status code and another set for the `307` status code.

The `/sys/config/ui` API endpoint allows users to set UI-specific custom headers. However, if the header is configured in the configuration file, it cannot be reconfigured through this endpoint. To remove or modify a custom header, you need to update the Stronghold configuration file and send the `SIGHUP` signal to the Stronghold process.

If a custom header is set in the configuration file and the same header is used by internal Stronghold processes, the configured header will not be accepted. For example, a custom header with the `X-Vault-` prefix will not be accepted. A corresponding log message will be registered in the Stronghold logs.

Priority Order

If the same header is specified both in the configuration file and via the `/sys/config/ui` API endpoint, the header from the configuration file takes priority. For example, the `Content-Security-Policy` header is defined by default in the `/sys/config/ui` endpoint. However, if the same header is specified in the configuration file, Stronghold will use it and substitute it in the response instead of the default value from `/sys/config/ui` .

TCP Listener Parameters

- `address` — string parameter in the format `ip-address:port`. Specifies the address and port to listen on. The parameter can be dynamically determined using the go-sockaddr template, which is resolved at runtime.
- `cluster_address` — string parameter in the format `ip-address:port`. Specifies the address and port to listen for server-to-server requests in the cluster. By default, the value of this parameter is one port higher than the `address` value. In most cases, you don't need to set `cluster_address`, but the parameter can be useful if isolated Stronghold servers need to skip a TCP load balancer or use another connection scheme. The parameter can be dynamically determined using the go-sockaddr template, which is resolved at runtime.
- `http_idle_timeout` — string parameter. Specifies the maximum time to wait for the next request when the keep-alive mode is enabled. The value is specified with a time suffix, such as `"40s"` (40 seconds) or `"1h"` (1 hour). If the parameter value is zero, the `http_read_timeout` value will be used. If both values are zero, the `http_read_header_timeout` value will be used.
- `http_read_header_timeout` — string parameter. Specifies the time allocated for reading the request headers. The value is specified with a time suffix, such as `"40s"` (40 seconds) or `"1h"` (1 hour).
- `http_read_timeout` — string parameter. Specifies the maximum duration to read the entire request, including headers and body. The value is specified with a time suffix, such as `"40s"` (40 seconds) or `"1h"` (1 hour).
- `http_write_timeout` — string parameter. Specifies the maximum duration to write the response. It is reset every time a new request header is read. By default, `0` means no time limit. The value is specified with a time suffix, such as `"40s"` (40 seconds) or `"1h"` (1 hour).
- `max_request_size` — integer parameter. Specifies the maximum allowed size for a request in bytes. By default, it is set to 32 MB if not specified or set to `0`. A value less than `0` disables the limit.
- `max_request_duration` — string parameter. Specifies the maximum time to process a request after which Stronghold will cancel it. This parameter overrides the `default_max_request_duration` for this listener. The value is specified with a time suffix, such as `"90s"` (90 seconds).
- `proxy_protocol_behavior` — string parameter. Enables PROXY protocol version 1 behavior for the listener. The following values are accepted:
  - `use_always` — always uses the client IP address.
  - `allow_authorized` — uses the client IP address if the source IP is in the list.
  - `proxy_protocol_authorized_addrs` — uses the source IP if it is not in the list.
  - `deny_unauthorized` — if the source IP address is not in the `proxy_protocol_authorized_addrs` list, traffic will be rejected.

- `proxy_protocol_authorized_addrs` — string parameter or array of strings. Specifies the list of allowed source IP addresses for use with the PROXY protocol. It is not required if `proxy_protocol_behavior` is set to `use_always`. When specified as a string, values should be comma-separated. The value `proxy_protocol_authorized_addrs` cannot be empty; at least one source IP address must be specified.
- `redact_addresses` — boolean parameter. Hides the values of `leader_address` and `cluster_leader_address` in the corresponding API responses when set to `true`.
- `redact_cluster_name` — boolean parameter. Hides the value of `cluster_name` in the corresponding API responses when set to `true`.
- `redact_version` — boolean parameter. Hides the values of `version` and `build_date` in the corresponding API responses when set to `true`.
- `tls_disable` — boolean parameter. Specifies whether TLS is disabled. Stronghold uses TLS by default, so it is necessary to explicitly disable TLS if insecure communication is not desired. Disabling TLS may cause some UI features to be disabled.
- `tls_cert_file` — string parameter. Specifies the path to the TLS certificate file. The file must be in PEM format. To configure the listener to use the CA certificate, the primary certificate and the CA certificate should be concatenated. The path specified in the `tls_cert_file` parameter is used when starting Stronghold. Changing this value during Stronghold operation will have no effect.
- `tls_key_file` — string parameter. Specifies the path to the private key file for the certificate. The file must be in PEM format. If the key file is encrypted, a passphrase will be required when starting the server. When reloading the configuration via `SIGHUP`, the passphrase between key files must remain the same. The path specified in the `tls_key_file` parameter is used when starting Stronghold. Changing this value during Stronghold operation will have no effect.
- `tls_min_version` — string parameter. Specifies the minimum TLS version supported by the listener. Accepted values: `"tls10"`, `"tls11"`, `"tls12"`, or `"tls13"`. TLS 1.1 and below (i.e., `tls10` and `tls11`) are considered insecure and not recommended.
- `tls_max_version` — string parameter. Specifies the maximum TLS version supported by the listener. Accepted values: `"tls10"`, `"tls11"`, `"tls12"`, or `"tls13"`. TLS 1.1 and below (i.e., `tls10` and `tls11`) are considered insecure and not recommended.
- `tls_cipher_suites` — string parameter. Specifies the list of supported cipher suites as a comma-separated list of values. A list of all available cipher suites can be found in the Go TLS documentation.

  Go uses the list specified in the `tls_cipher_suites` parameter only for TLSv1.2 and earlier versions. The order of ciphers is not important. To optimize `tls_cipher_suites`, set `tls_max_version` to `"tls12"` to prevent the negotiation of TLSv1.3. More details about this and other related TLS changes can be found in the official Go TLS post.

- `tls_require_and_verify_client_cert` — boolean parameter. Enables client authentication for the listener. The listener will require a presented client certificate, successfully verified against system CAs.
- `tls_client_ca_file` — string parameter. A PEM-encoded certificate file for the CA used for client authentication.
- `tls_disable_client_certs` — boolean parameter. Disables client authentication for the listener. By default, `false` — Stronghold requests client authentication certificates when available.

  > **Warning.** The `tls_disable_client_certs` and `tls_require_and_verify_client_cert` fields in the `listener` section are mutually exclusive. Ensure that both are not set to `true`. By default, client certificate verification is not required.

- `x_forwarded_for_authorized_addrs` — string parameter. Can be specified as a comma-separated list of values or as a JSON array. Specifies the list of allowed source IP addresses for use with the `X-Forwarded-For` header. Enables support for `X-Forwarded-For`.

  For example, Stronghold receives a connection from the load balancer IP `1.2.3.4`. Adding `1.2.3.4` to the `x_forwarded_for_authorized_addrs` parameter will ensure that the client IP address initiating the connection, e.g., `3.4.5.6`, appears in the `remote_address` field of the audit log. It is important that the load balancer sends the client IP address in the `X-Forwarded-For` header.
- `x_forwarded_for_hop_skips` — string parameter. Specifies the number of addresses to skip from the end of the hop list. For example, if the `X-Forwarded-For` header contains addresses `1.2.3.4, 2.3.4.5, 3.4.5.6, 4.5.6.7`, and the `x_forwarded_for_hop_skips` parameter is set to `"1"`, the client IP address used will be `3.4.5.6`.
- `x_forwarded_for_reject_not_authorized` — boolean parameter. When set to `false`, it allows ignoring the `X-Forwarded-For` header if it comes from an unauthorized address. In this case, the client connection will be used as is, instead of being rejected.
- `x_forwarded_for_reject_not_present` — boolean parameter. When set to `false`, it allows using the client address as is if the `X-Forwarded-For` header is missing or empty, instead of rejecting the client connection.
- `disable_replication_status_endpoints` — boolean parameter. When set to `true`, it disables replication status endpoints for this listener.

Telemetry parameters:

- `unauthenticated_metrics_access` — boolean parameter. When set to `true`, it allows unauthenticated access to the `/v1/sys/metrics` endpoint.

Profiling parameters:

- `unauthenticated_pprof_access` — boolean parameter. When set to `true`, it allows unauthenticated access to the `/v1/sys/pprof` endpoint.

Inflight requests logging parameters:

- `unauthenticated_in_flight_requests_access` — boolean parameter. When set to `true`, it allows unauthenticated access to the `/v1/sys/in-flight-req` endpoint.

Custom response headers parameters:

A list of mappings of type:

```
{
  "key1" = ["value1", "value 2", ...],
  "key2" = ["value1", "value 2", ...],
}
```

allows mapping default header names to an array of values. Default headers are set across all endpoints regardless of the status code value.

The list of mappings is of the type:

```
{
  "key1" = ["value1", "value 2", ...],
  "key2" = ["value1", "value 2", ...],
}
```

allows mapping header names to an array of values. The headers specified in this section are set for the specified status codes.

The list of mappings is of the type:

```
{
  "key1" = ["value1", "value2", ...],
  "key2" = ["value1", "value2", ...],
}
```

allows mapping header names to an array of values. The headers specified in this section are set for status codes that fall into the specified status code groups.

Example configuration for TCP listener section

Example 1. Specifying required parameters for TLS.

This demonstrates specifying the certificate and key for TLS.

```
listener "tcp" {
  tls_cert_file = "/etc/certs/tls.crt"
  tls_key_file  = "/etc/certs/tls.key"
}
```

Example 2. Listening on multiple interfaces.

This shows how to listen on a private interface and localhost for Stronghold.

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

listener "tcp" {
  address = "10.0.0.5:8200"
}

# Advertise the non-loopback interface
api_addr = "https://10.0.0.5:8200"
cluster_addr = "https://10.0.0.5:8201"
```

Example 3. Allowing unauthenticated access to metrics.

```
listener "tcp" {
  telemetry {
    unauthenticated_metrics_access = true
  }
}
```

Example 4. Allowing unauthenticated access to profiling.

```
listener "tcp" {
  profiling {
    unauthenticated_pprof_access = true
    unauthenticated_in_flight_request_access = true
  }
}
```

Example 5. Configuring custom HTTP response headers.

Operators can configure the `custom_response_headers` subsection in the `listener` section to add custom HTTP headers relevant to their applications.

```
listener "tcp" {
  custom_response_headers {
    "default" = {
      "Strict-Transport-Security" = ["max-age=31536000","includeSubDomains"],
      "Content-Security-Policy" = ["connect-src https://clusterA.vault.external/"],
      "X-Custom-Header" = ["Custom Header Default Value"],
    },
    "2xx" = {
      "Content-Security-Policy" = ["connect-src https://clusterB.vault.external/"],
      "X-Custom-Header" = ["Custom Header Value 1", "Custom Header Value 2"],
    },
    "301" = {
      "Strict-Transport-Security" = ["max-age=31536000"],
      "Content-Security-Policy" = ["connect-src https://clusterC.vault.external/"],
    },
  }
}
```

Examples of custom HTTP headers — `Strict-Transport-Security` and `Content-Security-Policy`. These can be configured to enhance the security of the application interacting with Stronghold endpoints. Vulnerability scanners often look for such security-related HTTP headers. It is also possible to configure application-specific custom headers, as shown with `X-Custom-Header` in the example above.

If a header is defined in multiple status code subsections, the header corresponding to the most specific response code will be returned. From the configuration example below, the `306` response will return the `Custom` header for 3xx, while `307` will return the `Custom` header for 307.

```
listener "tcp" {
  custom_response_headers {
    "default" = {
       "X-Custom-Header" = ["default Custom header value"]
    },
    "3xx" = {
       "X-Custom-Header" = ["3xx Custom header value"]
    },
    "307" = {
       "X-Custom-Header" = ["307 Custom header value"]
    }
  }
}
```

Example 6. Listening on all IPv4 and IPv6 interfaces

In this example, Stronghold listens on all IPv4 and IPv6 interfaces, including localhost.

```
listener "tcp" {
  address         = "[::]:8200"
  cluster_address = "[::]:8201"
}
```

Example 7. Listening on specific IPv6 addresses

In this example, it is configured to use only IPv6, bound to the interface with IP address `2001:1c04:90d:1c00:a00:27ff:fefa:58ec`.

```
listener "tcp" {
  address         = "[2001:1c04:90d:1c00:a00:27ff:fefa:58ec]:8200"
  cluster_address = "[2001:1c04:90d:1c00:a00:27ff:fefa:58ec]:8201"
}
```

Declaring a non-loopback interface:

```
api_addr = "https://[2001:1c04:90d:1c00:a00:27ff:fefa:58ec]:8200"
cluster_addr = "https://[2001:1c04:90d:1c00:a00:27ff:fefa:58ec]:8201"
```

Example of hiding information

Example 1. Configuration using `redact_addresses`, `redact_cluster_name`, and `redact_version` to hide information in responses.

```
ui            = true
cluster_addr  = "https://127.0.0.1:8201"
api_addr      = "https://127.0.0.1:8200"
disable_mlock = true

storage "raft" {
  path = "/path/to/raft/data"
  node_id = "raft_node_1"
}

listener "tcp" {
  address             = "127.0.0.1:8200",
  tls_cert_file = "/path/to/full-chain.pem"
  tls_key_file  = "/path/to/private-key.pem"
  redact_addresses    = "true"
  redact_cluster_name = "true"
  redact_version      = "true"
}

telemetry {
  statsite_address = "127.0.0.1:8125"
  disable_hostname = true
}
```

Example 2. Result of applying redaction parameters for `API: /sys/health`.

In the `/sys/health/` API call, the `cluster_name` is completely omitted from the response, and version is returned as an empty string (`""`).

```
$ curl -s https://127.0.0.1:8200/v1/sys/health | jq

{
  "initialized": true,
  "sealed": false,
  "standby": false,
  "performance_standby": false,
  "replication_performance_mode": "disabled",
  "replication_dr_mode": "disabled",
  "server_time_utc": 1715935559,
  "version": "",
  "cluster_id": "be574716-e7e9-a950-ee34-d62d56cd6d4a"
}
```

Example 3. Result of applying redaction parameters for `API: /sys/leader`.

In the `/sys/leader/` API call, the `leader_address` and `leader_cluster_address` are set to empty strings (`""`).

```
$ curl -s https://127.0.0.1:8200/v1/sys/leader | jq

{
  "ha_enabled": true,
  "is_self": true,
  "active_time": "2024-05-13T07:54:20.471072843Z",
  "leader_address": "",
  "leader_cluster_address": "",
  "performance_standby": false,
  "performance_standby_last_remote_wal": 0,
  "raft_committed_index": 78,
  "raft_applied_index": 78
}
```

Example 4. Result of applying redaction parameters for `API: /sys/seal-status`.

In the `/sys/seal-status/` API call, the `cluster_name`, `build_date`, and version fields are hidden. The `cluster_name` is completely omitted from the response, while `build_date` and version are returned as empty strings (`""`).

```
$ curl -s https://127.0.0.1:8200/v1/sys/seal-status | jq

{
  "type": "shamir",
  "initialized": true,
  "sealed": false,
  "t": 3,
  "n": 6,
  "progress": 0,
  "nonce": "",
  "version": "",
  "build_date": "",
  "migration": false,
  "cluster_id": "be574716-e7e9-a950-ee34-d62d56cd6d4a",
  "recovery_seal": false,
  "storage_type": "raft"
}
```

Example 5. CLI: `stronghold status`

The CLI command `stronghold status` uses endpoints that support data redaction, so the output hides `Version`, `Build Date`, `HA Cluster`, and `Active Node Address`.

`Version`, `Build Date`, and `HA Cluster` show `n/a` because the corresponding endpoint returned an empty string. On the other hand, `Active Node Address` is shown as `<none>` because the address was omitted in the API response.

```
stronghold status

Key                     Value
---                     -----
Seal Type               shamir
Initialized             true
Sealed                  false
Total Shares            6
Threshold               3
Version                 n/a
Build Date              n/a
Storage Type            raft
HA Enabled              true
HA Cluster              n/a
HA Mode                 active
Active Since            2024-05-13T07:54:20.471072843Z
Active Node Address     <none>
Raft Committed Index    78
Raft Applied Index      78
```

Unix

The Unix listener configuration configures Stronghold to listen on the specified Unix domain socket:

```
listener "unix" {
  address = "/run/vault.sock"
}
```

You can specify the `listener` section multiple times to have Stronghold listen on multiple sockets.

Unix listener parameters

- `address` — required string parameter. Specifies the address to bind the Unix socket.
- `socket_mode` — optional string parameter. Changes the permissions and special flags of the Unix socket.
- `socket_user` — optional string parameter. Changes the user owner of the Unix socket.
- `socket_group` — optional string parameter. Changes the group owner of the Unix socket.

Example Unix listener configuration

Example 1. Listening on multiple sockets.

In this example, Stronghold is configured to listen on the specified socket and the default socket.

```
listener "unix" {}

listener "unix" {
  address = "/var/run/vault.sock"
}
```

Example 2. Listening on multiple interfaces.

In this example, Stronghold is configured to listen on the specified Unix socket, as well as on the loopback interface.

```
listener "unix" {
  address = "/var/run/vault.sock"
}

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

Example 3. Configuring permissions.

This example shows the configuration of permissions and ownership — user and group.

```
listener "unix" {
  address = "/var/run/vault.sock"
  socket_mode = "644"
  socket_user = "1000"
  socket_group = "1000"
}
```

Storage section

The `storage` section configures the storage backend used for persistent data storage in Stronghold.

Each backend has its own advantages and disadvantages, with trade-offs. For example, some backends provide more reliable backup and restore processes, while others support high availability.

Configuration

The storage backend is configured in the Stronghold configuration file through the `storage` section.

```
storage [NAME] {
  [PARAMETERS...]
}
```

For example:

```
storage "file" {
  path = "/mnt/vault/data"
}
```

For configuration parameters that also read environment variables, the environment variable will take precedence over the values in the configuration file.

Filesystem storage backend

The filesystem storage backend stores Stronghold data in the file system using a standard directory structure. It can be used in scenarios with a single persistent server or for local development where durability is not critical.

The filesystem storage backend does not support high availability.

Example configuration of the `storage` section with this storage backend:

```
storage "file" {
  path = "/mnt/vault/data"
}
```

Although Stronghold data is encrypted at rest, appropriate measures must be taken to secure access to the file system.

Parameters

- `path` — required string parameter. The absolute path to the directory where data will be stored. If the directory does not exist, Stronghold will create it.

Raft storage Backend

The integrated storage backend (Raft) is used to store Stronghold data. Unlike other storage backends, it does not operate with a single data source; instead, all nodes in the Stronghold cluster will have a replicated copy of the data. Data is replicated across all nodes using the Raft consensus algorithm.

The Raft storage backend supports high availability.

Example configuration of the `storage` section with a Raft storage backend:

```
storage "raft" {
  path = "/path/to/raft/data"
  node_id = "raft_node_1"
}
cluster_addr = "http://127.0.0.1:8201"
```

When using the integrated storage backend, you must specify the `cluster_addr` parameter, which represents the address and port used for communication between nodes in the Raft cluster.

Additionally, when using the integrated storage backend, you cannot declare a separate `ha_storage` backend. It is also strongly recommended to set `disable_mlock` to `true` and disable swapping on the system.

Parameters

- `path` — string parameter. Specifies the path to the directory in the file system where Stronghold data is stored. This value can be overridden by setting the `VAULT_RAFT_PATH` environment variable.
- `node_id` — string parameter. The node identifier in the Raft cluster. This value can be overridden by setting the `VAULT_RAFT_NODE_ID` environment variable.
- `performance_multiplier` — integer parameter. A multiplier used by servers to scale key Raft timing parameters.

  The `performance_multiplier` setting affects the time Stronghold needs to detect leader failures and elect a leader. This is achieved by requiring more network and computational resources to improve performance. If the value is not set or set to `0`, the default time described below will be used. Lower values tighten the time and increase sensitivity, while higher values loosen the time and reduce sensitivity.

  By default, Stronghold uses lower-performance timing parameters, suitable for servers meeting the minimum Stronghold requirements. Currently, this is equivalent to setting the value to `5`, but the standard may change in future versions of Stronghold if the minimum server profile changes. Setting `performance_multiplier` to `1` configures Raft for the highest performance mode, which is recommended for Stronghold servers in production environments. The maximum allowed value is `10`.
- `trailing_logs` — integer parameter. Controls the number of log entries that remain in the log storage on disk after a snapshot is created. This parameter should only be adjusted if followers cannot catch up with the leader due to a very large snapshot size and high write throughput, which causes log truncation before the snapshot can be fully applied.

  If you need to use this for cluster recovery, consider lowering the write throughput or the volume of data stored in Stronghold. The default value is `10000`, which is suitable for most normal workloads. The `trailing_logs` metric is not equivalent to the `max_trailing_logs` parameter.
- `snapshot_threshold` — integer parameter. Controls the minimum number of Raft log entries between snapshots that are saved to disk. Typically, this low-level parameter does not require modification. In highly loaded clusters with excessive disk I/O, increasing the value can reduce disk load and minimize the likelihood of simultaneous snapshot creation on all servers.

  Increasing the `snapshot_threshold` parameter is a trade-off between disk I/O and disk space, as the Raft log will grow significantly, and the space in the `raft.db` file will not be cleared until the next snapshot. Also, during failure recovery or failover, servers will require more time, as they will need to replay more logs. The default value is `8192`.

- `snapshot_interval` — integer parameter. The interval between snapshots in seconds, controlling how often Raft checks whether a snapshot is needed. To avoid the entire cluster creating a snapshot at the same time, Raft randomly selects a time between the specified interval and its double. By default, this is set to 120 seconds.
- `retry_join` — list parameter. Allows configuring a set of connection parameters for other nodes in the cluster. This set helps nodes find the leader to join the cluster. Multiple `retry_join` sections can be defined.

  If the connection parameters for all nodes in the cluster are known in advance, these sections can be enabled. In this case, once one node is initialized as the leader, the others will use their `retry_join` configuration to find the leader and join the cluster. Note that when using Shamir Secret Sharing, the joined nodes will need to be manually unlocked.

  For more information on the `retry_join` section parameters, see retry_join section parameters.
- `retry_join_as_non_voter` — boolean parameter. When set to `true`, any `retry_join` configuration will join the Raft cluster as a non-voting member. The node will not participate in the Raft quorum but will receive the data replication stream. This allows the cluster to scale reading.

  This option has the same effect as the `-non-voter` flag for the `stronghold operator raft join` command but only affects the voting status when joining via the `retry_join` configuration. The parameter can be overridden by setting the `VAULT_RAFT_RETRY_JOIN_AS_NON_VOTER` environment variable to any non-empty value. This only applies if at least one `retry_join` section is specified.
- `max_entry_size` — integer parameter. Configures the maximum number of bytes for a Raft entry. This applies to both put operations and transactions. Any put operation or transaction that exceeds this configuration value will cause the corresponding operation to fail. Raft has a recommended maximum data size for an entry in the log, based on the current architecture, standard timing, etc.

  The integrated storage also uses a block size — a threshold applied to split large values into blocks. By default, the block size equals the maximum size of a Raft log entry. The default value is `1048576`.
- `autopilot_reconcile_interval` — string parameter. Specifies the time interval after which the autopilot will detect any state changes.

  State changes can indicate various things, such as:

  1. A node, initially added as a non-voting node in the Raft cluster, has successfully completed the stabilization period, qualifying it to be promoted to voting status.
  2. A node should be marked as `unhealthy` in the state API.
  3. A node has been marked as `dead` and should be removed from the Raft configuration.

  The value is specified with a time suffix, for example, `"40s"` (40 seconds) or `"1h"` (1 hour).

- `autopilot_update_interval` — string parameter. Specifies the time interval after which the autopilot will query Stronghold for updates on the relevant information. This includes data such as autopilot configuration and current state, Raft configuration, known servers, the latest Raft index, and statistics for all known servers. The retrieved information will be used to calculate the autopilot's next state. The value is specified with a time suffix, for example, `"40s"` (40 seconds) or `"1h"` (1 hour).

Parameters for `retry_join` section

- `leader_api_addr` — string parameter. The IP address of a potential leader node.
- `leader_tls_servername` — string parameter. The TLS server name used when connecting via HTTPS. It must match one of the names in the DNS SANs (Subject Alternative Names) of the leader's TLS certificate. The node uses `leader_tls_servername` to verify the leader's certificate when attempting to connect to the leader of the cluster. This ensures secure connections and verifies that the connection is made with the correct server.
- `leader_ca_cert_file` — string parameter. Path to the CA certificate file of the potential leader node.
- `leader_client_cert_file` — string parameter. Path to the client certificate file used for authentication when connecting to the leader via TLS. The Raft node presents this certificate when establishing a secure connection with the leader to authenticate itself.
- `leader_client_key_file` — string parameter. Path to the private key file used in conjunction with the client certificate for authentication when connecting to the leader via TLS.
- `leader_ca_cert` — string parameter. The CA certificate value of the potential leader node.
- `leader_client_cert` — string parameter. The client certificate value used for authentication when connecting to the leader via TLS. The Raft node presents this certificate when establishing a secure connection with the leader to authenticate itself.
- `leader_client_key` — string parameter. The private key value used in conjunction with the client certificate for authentication when connecting to the leader via TLS.

Each `retry_join` block can provide TLS certificates either through file paths or as certificate values, but not a combination of both. If a certificate value is provided, it must be specified in a single line using `\n` to denote required line breaks.

UI section

Stronghold provides a user interface (web interface) for operators. It enables easy creation, reading, updating, and deletion of secrets, authentication, storage unsealing, and much more.

Enabling UI

By default, the Stronghold user interface is not enabled. To enable it, set the `ui` parameter in the Stronghold server configuration to `true`.

```
ui = true

listener "tcp" {
  # ...
}
```

Accessing Stronghold UI

The user interface runs on the same port as the Stronghold listener. To access the UI, at least one `listener` section must be configured.

```
listener "tcp" {
  address = "10.0.1.35:8200"
  ...
}
```

The user interface in the example provided is accessible via the URL https://10.0.1.35:8200/ui/ from any machine in the subnet — assuming there are no firewalls or the firewalls are properly configured. The UI is also accessible via any DNS entry that resolves to this IP address.

user_lockout section

The `user_lockout` section defines the settings for locking users after unsuccessful login attempts to Stronghold. These settings can be applied globally — for all authentication methods (userpass, ldap, and approle) using the common section name `user_lockout "all"`, or individually for specific methods by specifying their name in the section. Supported values are: `all`, `userpass`, `ldap`, and `approle`.

Configurations specified for a specific authentication method take precedence over settings for all authentication methods using the `user_lockout "all"` section. If both configurations are present, the parameters for the specific method will be applied.

Parameters for the `user_lockout` section

- `lockout_threshold` — string parameter. Specifies the number of failed login attempts after which the user will be locked out.
- `lockout_duration` — string parameter. Specifies the duration of the user's lockout. The value is specified with a time suffix, for example, `"40s"` (40 seconds) or `"1h"` (1 hour).
- `lockout_counter_reset` — string parameter. Defines the time interval after which the lockout counter is reset if no failed login attempts occur. The value is specified with a time suffix, for example, `"40s"` (40 seconds) or `"1h"` (1 hour).
- `disable_lockout` — boolean parameter. Disables the user lockout feature if set to `true`.

Recover from lost quorum

Quorum is the minimum number of nodes in a cluster required to vote and elect a leader. A Raft leader is the active cluster node that performs read and write operations and coordinates the other nodes.

With Integrated Storage, maintaining Raft quorum is an important factor when configuring and operating a Stronghold environment with HA enabled. A Stronghold cluster permanently loses quorum when there is no way to recover enough servers to reach consensus and elect a leader. Without a quorum of cluster servers, Stronghold can no longer perform read and write operations.

The cluster quorum is dynamically updated when new servers join the cluster. Stronghold calculates quorum with the formula `(n+1)/2`, where `n` is the number of servers in the cluster. For example, for a 3-server cluster, you will need at least 2 servers operational for the cluster to function properly, `(3+1)/2 = 2`. Specifically, you will need 2 servers always active to perform read and write operations.

> There is an exception to this rule if you use the `-non-voter` option while joining the cluster. This feature is available only in Stronghold as a standalone.

Quorum loss scenario

When two of the three servers become unavailable, the cluster loses quorum and becomes inoperable.

Although one of the servers is fully functioning, the cluster won't be able to process read or write requests.

Example:

1. Response from console:

```
$ stronghold operator raft list-peers
* local node not active but active cluster node not found

$ stronghold kv get kv/apikey
* local node not active but active cluster node not found
```

2. Logs from one of inoperative nodes:

```
oct 20 10:54:32 standalone-astra stronghold[647]:
{"@level":"info","@message":"attempting to join possible raft leader
node","@module":"core","@timestamp":"2025-10-20T10:54:02.578963Z","leader_addr":
"https://stronghold-0.stronghold.tld:8201"}
oct 20 10:54:32 standalone-astra stronghold[647]:
{"@level":"error","@message":"failed to get raft
challenge","@module":"core","@timestamp":"2025-10-20T10:54:32.597558Z","error":"
error during raft bootstrap init call: Put \"https://10.0.101.22:8201/v1/sys/
storage/raft/bootstrap/challenge\": dial tcp 10.0.101.22:8201: i/o
timeout","leader_addr":"https://stronghold-0.stronghold.tld:8201"}
```

In this tutorial, you will recover from the permanent loss of two-of-three Stronghold servers by converting it into a single-server cluster.

The remaining server must be fully operational to complete this procedure.

In a 5-server cluster, or when non-voters are present, you must stop all other healthy servers before performing the peers.json recovery.

Autopilot recovery considerations

Autopilot is a Stronghold mechanism that automatically monitors the state of Raft cluster nodes and manages their participation in quorum.

In some cases, Stronghold may lose quorum due to Autopilot marking servers as unhealthy, while the service is still running. In such situations, before performing recovery using peers.json, you must stop Stronghold services on the unhealthy servers.

Recover from lost quorum

Locate the storage directory

On the healthy Stronghold server, locate the Raft storage directory. To discover the location of the directory, review your Stronghold configuration file. The `storage` stanza will contain the `path` to the directory.

Example:

`/opt/stronghold/config.hcl`

```
storage "raft" {
  path    = "/opt/stronghold/data"
  server_id = "stronghold_0"
}

listener "tcp" {
  address     = "0.0.0.0:8200"
  cluster_address     = "0.0.0.0:8201"
  tls_disable = true
}

api_addr = "http://stronghold-0.stronghold.tld:8200"
cluster_addr = "http://stronghold-0.stronghold.tld:8201"
disable_mlock = true
ui=true
```

In this example, the `path` is the file system path where Stronghold stores data, and the `server_id` is the identifier for the server in the Raft cluster. The example `server_id` is `stronghold_0`.

Create the peers.json file

Inside the storage directory (`/opt/stronghold/data`), there is a folder named `raft`.

```
/opt
└ stronghold
  └── data
      ├── raft
      │   ├── raft.db
      │   └── snapshots
      └── vault.db
```

To enable the single, remaining Stronghold server to reach quorum and elect itself as the leader, create a `raft/peers.json` file that holds the server information. The file format is a JSON array containing the server ID, *address:port*, and suffrage information of the healthy Stronghold server.

Example:

```
$ cat > /stronghold/data/raft/peers.json << EOF
[
  {
    "id": "stronghold_0",
    "address": "stronghold-0.stronghold.tld:8201",
    "non_voter": false
  }
]
EOF
```

- **id** (string: <required>) - Specifies the server ID of the server.
- **address** (string: <required>) - Specifies the host and port of the server. The port is the server's cluster port.
- **non_voter** (bool: <false>) - This controls whether the server is a non-voter.

Make sure `stronghold` user has *read* and *edit* permissions on `peers.json` file.

```
chown stronghold:stronghold /opt/stronghold/data/raft/peers.json
chmod 600 /opt/stronghold/data/raft/peers.json
```

Restart Stronghold

Restart the Stronghold process to enable Stronghold to load the new `peers.json` file.

```
sudo systemctl restart stronghold
```


f you use Systemd, a `SIGHUP` signal will not work.

Unseal Stronghold

If not configured to use auto-unseal, unseal Stronghold and then check the status.

Example:

```
$ stronghold operator unseal
Unseal Key (will be hidden):

$ stronghold status
Key                     Value
---                     -----
Recovery Seal Type      shamir
Initialized             true
Sealed                  false
Total Recovery Shares   1
Threshold               1
Version                 1.16.0+hsm
Storage Type            raft
Cluster Name            stronghold-cluster-4a1a40af
Cluster ID              d09df2c7-1d3e-f7d0-a9f7-93fadcc29110
HA Enabled              true
HA Cluster              https://stronghold-0.stronghold.tld:8201
HA Mode                 active
Active Since            2021-07-20T00:07:32.215236307Z
Raft Committed Index    155344
Raft Applied Index      155344
```

Verify recovery success

The recovery procedure is successful when Stronghold starts up and displays these messages in the system logs.

```
...snip...
[INFO]  core.cluster-listener: serving cluster requests: cluster_listen_address=[::]:
8201
[INFO]  storage.raft: raft recovery initiated: recovery_file=peers.json
[INFO]  storage.raft: raft recovery found new config: config="{[{Voter stronghold_0
https://stronghold-0.stronghold.tld:8201}]}"
[INFO]  storage.raft: raft recovery deleted peers.json
...snip...
```

View the peer list

You now have a cluster with one server that can reach the quorum. Verify that there is just one server in the cluster with `stronghold operator raft list-peers` command.

```
$ stronghold operator raft list-peers
Node            Address                                 State     Voter
----            -------                                 -----     -----
stronghold_0    https://stronghold-0.stronghold.tld:8201    leader    true
```

Next steps

In this tutorial, you recovered the loss of quorum by converting a 3-server cluster into a single-server cluster using the `peers.json`. The `peers.json` file enabled you to manually overwrite the Raft peer list to the one remaining server, which allowed that server to reach quorum and complete a leader election.

If the failed servers are **recoverable**, the best option is to bring them back online and have them reconnect to the cluster using the same host addresses. This will return the cluster to a fully healthy state. In such an event, the `raft/peers.json` should contain the server ID, *address:port*, and suffrage information of each Stronghold server you wish to be in the cluster.

```
[
  {
    "id": "stronghold_0",
    "address": "stronghold-0.stronghold.tld:8201",
    "non_voter": false
  },
  {
    "id": "stronghold_1",
    "address": "stronghold-1.stronghold.tld:8201",
    "non_voter": false
  },
  {
    "id": "stronghold_2",
    "address": "stronghold-2.stronghold.tld:8201",
    "non_voter": false
  }
]
```

Launch in DKP

Release channels

Deckhouse Stronghold uses five release channels intended for use in different environments, with varying requirements in terms of stability:

Release channel	Description
Alpha	The least stable release channel but introduces new versions most frequently. Intended for development clusters used by a small number of developers.
Beta	Intended for development clusters, similar to the Alpha channel. Receives versions that have been previously tested on the Alpha channel.
Early Access	The recommended release channel if you are unsure which channel to use. Suitable for clusters with active development, such as running applications or debugging. New features are introduced to this channel no earlier than one week after their initial release.
Stable	The stable release channel for clusters where active development has finished and the focus is on normal operation. New features are introduced to this channel no earlier than two weeks after their initial release.
Rock Solid	The most stable release channel. Suitable for clusters that require a higher level of stability. New features are introduced to this channel no earlier than one month after their initial release.

Deckhouse Stronghold components can update either automatically or upon manual confirmation as updates are released in the release channels.

For information about the versions available in release channels, visit releases.deckhouse.io.

For details on configuring release channels, refer to Platform update.

Platform installation

Installation steps

Prepare the environment

Before installing the platform, please complete the following steps to prepare the infrastructure:

- Configure the servers:
  - Install a supported operating system.
  - Set up SSH access using key authentication.
- Ensure access to the container registry with Deckhouse images (by default, `registry.deckhouse.io`).

Platform installation

Configuration Preparation

To install the platform, you need to prepare a YAML configuration file for installation and, if necessary, a YAML file for resources that need to be created after a successful installation of the platform.

Installation Configuration File

The YAML installation configuration file includes parameters for several resources (manifests):

- InitConfiguration — initial parameters for the platform configuration. The platform will launch after installation with this configuration. This resource specifies parameters necessary for the platform to start and operate correctly, such as component placement parameters, the used StorageClass, container registry access settings, the template for DNS names, and others.
- ClusterConfiguration — general parameters of the cluster, such as control plane version, network parameters, CRI settings, etc.

  > You need to use the `ClusterConfiguration` resource only if you need to pre-deploy a Kubernetes cluster during the platform installation. `ClusterConfiguration` is not needed if the platform is installed in an existing Kubernetes cluster.

- StaticClusterConfiguration — parameters for a Kubernetes cluster deployed on bare metal servers or virtual machines in unsupported clouds.

  > Similar to `ClusterConfiguration`, `StaticClusterConfiguration` is not needed if the platform is installed in an existing Kubernetes cluster.

- ModuleConfig — a set of resources containing configuration parameters for built-in platform modules.

  If the cluster is initially created with nodes designated for specific types of workloads (system nodes, monitoring nodes, etc.), it is recommended to explicitly specify the corresponding nodeSelector in the module configuration for modules using persistent storage volumes (e.g., for the `prometheus` module, this would be the nodeSelector parameter).

Example Installation Configuration File…

```
apiVersion: deckhouse.io/v1
kind: ClusterConfiguration
clusterType: Cloud
cloud:
  provider: Azure
  prefix: cloud-demo
podSubnetCIDR: 10.111.0.0/16
serviceSubnetCIDR: 10.222.0.0/16
kubernetesVersion: "Automatic"
clusterDomain: cluster.local
---
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: deckhouse
spec:
  enabled: true
  settings:
    releaseChannel: Stable
    bundle: Default
    logLevel: Info
  version: 1
---
apiVersion: deckhouse.io/v1
kind: AzureClusterConfiguration
layout: Standard
sshPublicKey: <SSH_PUBLIC_KEY>
vNetCIDR: 10.241.0.0/16
subnetCIDR: 10.241.0.0/24
masterNodeGroup:
  replicas: 3
  instanceClass:
    machineSize: Standard_D4ds_v4
    urn: Canonical:UbuntuServer:18.04-LTS:18.04.202010140
    enableExternalIP: true
provider:
  subscriptionId: <SUBSCRIPTION_ID>
  clientId: <CLIENT_ID>
  clientSecret: <CLIENT_SECRET>
  tenantId: <TENANT_ID>
  location: westeurope
---
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: cni-flannel
spec:
  enabled: true
```

```
---
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: global
spec:
  enabled: true
  settings:
    modules:
      publicDomainTemplate: "%s.k8s.example.com"
  version: 1
---
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: node-manager
spec:
  version: 1
  enabled: true
  settings:
    allowedBundles: ["ubuntu-lts"]
---
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: prometheus
spec:
  version: 2
  enabled: true
  # Specify if using dedicated monitoring nodes.
  # settings:
  #   nodeSelector:
  #     node.deckhouse.io/group: monitoring
```

Installation Resources File

An optional YAML file of installation resources contains Kubernetes resource manifests that the installer will apply after the successful installation of the platform. This file can be useful for additional configuration of the cluster post-installation: deploying an Ingress controller, creating additional node groups, configuring resources, setting permissions, users, etc.

Attention! You cannot use ModuleConfig for **built-in** modules in the installation resources file. Use the configuration file for configuring built-in modules.

Example Installation Resources File…

```
---
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: global
spec:
  version: 1apiVersion: deckhouse.io/v1
kind: IngressNginxController
metadata:
  name: main
spec:
  ingressClass: "nginx"
  controllerVersion: "1.1"
  inlet: "LoadBalancer"
  nodeSelector:
    node.deckhouse.io/group: worker
---
apiVersion: deckhouse.io/v1
kind: ClusterAuthorizationRule
metadata:
  name: admin
spec:
  subjects:
  - kind: User
    name: admin@deckhouse.io
  accessLevel: SuperAdmin
  portForwarding: true
---
apiVersion: deckhouse.io/v1
kind: User
metadata:
  name: admin
spec:
  email: admin@deckhouse.io
  password: '$2a$10$isZrV6uzS6F7eGfaNB1EteLTWky7qxJZfbogRs1egWEPuT1XaOGg2'
---
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: deckhouse-admin
spec:
  enabled: true
  settings:
    modules:
      publicDomainTemplate: "%s.example.com"
      https:
        certManager:
          clusterIssuerName: selfsigned
```

```
        mode: CertManager
---
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: user-authn
spec:
  version: 2
  enabled: true
  settings:
    controlPlaneConfigurator:
      dexCAMode: FromIngressSecret
---
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: stronghold
spec:
  enabled: true
  version: 1
  settings:
    management:
      mode: Automatic
      administrators:
      - type: Group
        name: admins
---
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: secrets-store-integration
spec:
  enabled: true
  version: 1
---
apiVersion: deckhouse.io/v1
kind: IngressNginxController
metadata:
  name: main
spec:
  inlet: HostPort
  enableIstioSidecar: true
  ingressClass: nginx
  hostPort:
    httpPort: 80
    httpsPort: 443
  nodeSelector:
    node-role.kubernetes.io/master: ''
  tolerations:
```

```
    - effect: NoSchedule
      operator: Exists
---
apiVersion: deckhouse.io/v1
kind: ClusterAuthorizationRule
metadata:
  name: admin
spec:
  subjects:
  - kind: User
    name: admin@deckhouse.io
  accessLevel: SuperAdmin
  portForwarding: true
---
apiVersion: deckhouse.io/v1
kind: User
metadata:
  name: admin
spec:
  email: admin@deckhouse.io
  password: '$2a$10$isZrV6uzS6F7eGfaNB1EteLTWky7qxJZfbogRs1egWEPuT1XaOGg2'
---
apiVersion: deckhouse.io/v1alpha1
kind: Group
metadata:
  name: admins
spec:
  name: admins
  members:
  - kind: User
    name: admin
```

Post-Bootstrap Script

After a successful platform installation, the installer can run a script on one of the master nodes. This script can be used for additional setup, collecting setup information, etc.

You can specify a post-bootstrap script with the `--post-bootstrap-script-path` parameter during the installation run (see below).

Example of a Script Displaying the Load Balancer's IP Address…

Example of a script that displays the load balancer's IP address after the cluster deployment in the cloud and the platform installation:

```
#!/usr/bin/env bash
set -e
set -o pipefail

INGRESS_NAME="nginx"

echo_err() { echo "$@" 1>&2; }
# declare the variable
lb_ip=""
# get the load balancer IP
for i in {0..100}
do
  if lb_ip="$(d8 k -n d8-ingress-nginx get svc "${INGRESS_NAME}-load-balancer" -o json
path='{.status.loadBalancer.ingress[0].ip}')"; then
    if [ -n "$lb_ip" ]; then
      break
    fi
  fi
  lb_ip=""
  sleep 5
done
if [ -n "$lb_ip" ]; then
  echo_err "The load balancer external IP: $lb_ip"
else
  echo_err "Could not get the external IP of the load balancer"
  exit 1
fi
outContent="{\"frontend_ips\":[\"$lb_ip\"]}"
if [ -z "$OUTPUT" ]; then
  echo_err "The OUTPUT env is empty. The result was not saved to the output file."
else
  echo "$outContent" > "$OUTPUT"
fi
```

Platform Installation

> When installing a platform edition other than the Community Edition from the official container registry `registry.deckhouse.io`, you need to authenticate with a license key beforehand:
>
> ```
> docker login -u license-token registry.deckhouse.io
> ```

In general, the command to run the installer container from the platform's public container registry is as follows:

```
docker run --pull=always -it [<MOUNT_OPTIONS>] registry.deckhouse.io/deckhouse/
<DECKHOUSE_REVISION>/install:<RELEASE_CHANNEL> bash
```

where:

- `<DECKHOUSE_REVISION>` — edition of the platform (e.g., `ee` for Enterprise Edition, `ce` for Community Edition, etc.)
- `<MOUNT_OPTIONS>` — options for mounting files into the installer container, such as:
  - SSH access keys;
  - configuration file;
  - resources file, etc.
- `<RELEASE_CHANNEL>` — release channel of the platform in kebab-case. It should match the one set in `config.yaml`:
  - `alpha` — for the *Alpha* release channel;
  - `beta` — for the *Beta* release channel;
  - `early-access` — for the *Early Access* release channel;
  - `stable` — for the *Stable* release channel;
  - `rock-solid` — for the *Rock Solid* release channel.

Example of running the installer container for the CE edition:

```
docker run -it --pull=always \
  -v "$PWD/config.yaml:/config.yaml" \
  -v "$PWD/resources.yaml:/resources.yaml" \
  -v "$PWD/dhctl-tmp:/tmp/dhctl" \
  -v "$HOME/.ssh/:/tmp/.ssh/" registry.deckhouse.io/deckhouse/ce/install:stable bash
```

The platform installation is initiated in the installer container using the `dhctl` command:

- Use the `dhctl bootstrap` command to start the platform installation with cluster deployment (this applies to all cases except installation in an existing cluster).
- Use the `dhctl bootstrap-phase install-deckhouse` command to install the platform in an existing cluster.

> For help on the parameters, run `dhctl bootstrap -h`.

Example of starting a platform installation with cluster deployment in the cloud:

```
dhctl bootstrap \
  --ssh-user=<SSH_USER> --ssh-agent-private-keys=/tmp/.ssh/id_rsa \
  --config=/config.yaml --config=/resources.yaml
```

where:

- `/config.yaml` — installation configuration file;
- `/resources.yaml` — resource manifests file;
- `<SSH_USER>` — user on the server for SSH connection;
- `--ssh-agent-private-keys` — file containing the private SSH key for the connection.

Next, connect to the master node via SSH (the master node's IP address is provided by the installer at the end of the installation):

```
ssh <USER_NAME>@<MASTER_IP>
```

Starting the Ingress controller after the platform installation may take some time. Ensure the Ingress controller is running before proceeding:

```
d8 k -n d8-ingress-nginx get po
```

Wait until the Pods reach the status `Ready`.

Also, wait for the load balancer's readiness:

```
d8 k -n d8-ingress-nginx get svc nginx-load-balancer
```

The `EXTERNAL-IP` should be populated with a public IP address or DNS name.

DNS Configuration

To access the web interfaces of platform components, you need to:

1. Set up DNS.
2. Specify a DNS name template in the platform parameters.

The DNS name template is used for configuring Ingress resources of system applications. For example, the Grafana interface is assigned the name `grafana`. Thus, for the template `%s.kube.company.my`, Grafana will be accessible at `grafana.kube.company.my`, etc.

To simplify the setup, the `sslip.io` service will be used.

On the master node, execute the following command to obtain the load balancer's IP address and configure the DNS name template for platform services to use `sslip.io`:

```
BALANCER_IP=$(d8 k -n d8-ingress-nginx get svc nginx-load-balancer -o json | jq -r '.s
tatus.loadBalancer.ingress[0].ip') && \
echo "Balancer IP is '${BALANCER_IP}'." && d8 k patch mc global --type merge \
  -p "{\"spec\": {\"settings\":{\"modules\":{\"publicDomainTemplate\":\"%s.${BALANCER_
IP}.sslip.io\"}}}}" && echo && \
echo "Domain template is '$(d8 k get mc global -o=jsonpath='{.spec.settings.modules.pu
blicDomainTemplate}')'."
```

The command will also display the installed DNS name template. Example output:

```
Balancer IP is '1.2.3.4'.
moduleconfig.deckhouse.io/global patched
Domain template is '%s.1.2.3.4.sslip.io'.
```

Initial access configuration

After the installation is complete, you can connect to the platform in the following ways:

- Directly from the master node
- Remotely from a pre-configured personal computer

Connecting to the Platform from the Master Node

Connect to the master node via SSH (the master node's IP address is provided by the installer at the end of the installation):

```
ssh <USER_NAME>@<MASTER_IP>
```

Verify that platform resources are accessible by listing the cluster nodes:

```
d8 k get nodes
```

Remote Connection to the Platform

You can set up a remote connection to the cluster. To do this, follow the steps on your personal computer as described in the instructions.

Platform configuration

Node management

Node configuration

OS configuration

Coming soon…

Containerd configuration

Coming soon…

Node configuration

Coming soon…

Node groups

Coming soon…

Adding a node

Coming soon…

Advanced configuration

Coming soon…

Control plane

Components

Description

Coming soon…

Overview

Coming soon…

Etcd operations

Coming soon…

Scheduler

Coming soon…

Audit

Coming soon…

Master nodes

Coming soon…

Network management

Incoming traffic management

To provide external access to virtual machines, for example, for service publishing or remote administration, you can use `Ingress` resources, which are managed by the ingress-nginx module.

These created `Ingress` resources use Nginx as a reverse proxy and load balancer. If the cluster includes multiple nodes for hosting the Ingress controller, it will be deployed in failover mode, enhancing access reliability and resilience to failures.

Multiple instances of the Ingress NGINX Controller can run with separate configurations: one primary controller and any number of additional controllers. This approach lets you separate handling of `Ingress` resources for external and internal (intranet) applications, ensuring their isolation and more flexible access control.

Create a controller

To create a Ingress NGINX Controller, apply the following IngressNginxController resource:

```
d8 k apply -f - <<EOF
apiVersion: deckhouse.io/v1
kind: IngressNginxController
metadata:
  name: main
spec:
  ingressClass: nginx
  inlet: HostWithFailover
  nodeSelector:
    node-role.deckhouse.io/frontend: ""
  tolerations:
    - effect: NoExecute
      key: dedicated.deckhouse.io
      value: frontend
EOF
```

For configuration details on the `IngressNginxController` resource, refer to the corresponding article.

HTTPS termination

The `ingress-nginx` module lets you configure HTTPS security policies for each Ingress NGINX Controller, including:

- Managing HTTP Strict Transport Security (HSTS) parameters
- Configuring supported versions of SSL/TLS and encryption protocols

Also, ingress-nginx is integrated with the cert-manager module, which can be used for automatic ordering of SSL certificates and their use by the controllers.

Monitoring and statistics

The DVP `ingress-nginx` implementation includes a statistics collection system in Prometheus, providing various metrics:

- Overall response time and upstream time
- Response codes
- Number of request retries
- Request and response sizes
- Request methods
- `content-type` types
- Geographic distribution of requests

Data is displayed in multiple views:

- By namespace
- By `vhost`
- By Ingress resource
- By `location` (in NGINX)

All graphs are available on user-friendly dashboards in Grafana. You can drill down into the data. For example, when viewing statistics by namespace, you can explore deeper statistics by `vhosts` within that namespace by clicking the corresponding link on the Grafana dashboard.

Basic principles of statistics collection

1. At the `log_by_lua_block` stage, a module is called for each request to calculate and store the required data in a buffer. Each NGINX worker has its own buffer.
2. At the `init_by_lua_block` stage, a process is started for each NGINX worker. This process asynchronously sends data in `protobuf` format via a TCP socket to `protobuf_exporter` once per second.
3. `protobuf_exporter` runs as a sidecar container in the pod with the Ingress controller. It receives, parses, and aggregates the `protobuf` messages according to predefined rules, and then exports the data in a format compatible with Prometheus.

4. Prometheus scrapes metrics every 30 seconds from both the Ingress controller, which collects a small number of necessary metrics, and `protobuf_exporter`. This ensures efficient system performance.

Statistics collection and representation

All collected metrics have service labels visible in `/prometheus/targets` that let you identify the controller instance: controller, app, instance, and endpoint.

- All metrics exported by protobuf_exporter (except for geo) are represented in three levels of detail:
  - `ingress_nginx_overall_*`: An overview. All metrics have the `namespace`, `vhost`, and `content_kind` labels.
  - `ingress_nginx_detail_*`: In addition to the overall labels, this level adds `ingress`, `service`, `service_port`, and `location` labels.
  - `ingress_nginx_detail_backend_*`: A limited part of the data collected on a per-backend basis. In addition to the detail labels, these metrics have the `pod_ip` label.
- The following metrics are collected for the overall and detail levels:
  - `*_requests_total`: A request number counter. Additional labels: `scheme`, `method`.
  - `*_responses_total`: A response number counter. Additional label: `status`.
  - `*_request_seconds_{sum,count,bucket}`: A response time histogram.
  - `*_bytes_received_{sum,count,bucket}`: A request size histogram.
  - `*_bytes_sent_{sum,count,bucket}`: A response size histogram.
  - `*_upstream_response_seconds_{sum,count,bucket}`: An upstream response time histogram. If there were multiple upstreams, it tracks the sum of response times of all upstreams.
  - `*_lowres_upstream_response_seconds_{sum,count,bucket}`: The same as the previous metric but less detailed. It's suitable for visualization but not for quantile calculation.
  - `*_upstream_retries_{count,sum}`: A number of requests with backend retries and the sum of retries.
- The following metrics are collected for the overall level:
  - `*_geohash_total`: A counter for the number of requests with a specific geohash. Additional labels: `geohash`, `place`.
- The following metrics are collected for the detail_backend level:
  - `*_lowres_upstream_response_seconds`: The same metric as the one used for the overall and detail levels.
  - `*_responses_total`: A response number counter. Additional label: `status_class`.
  - `*_upstream_bytes_received_sum`: A counter for the total size of backend responses.

Outgoing traffic management

⚠ This feature isn’t available in Community Edition.

In a deployed cluster, some nodes may be isolated from the external network for security reasons. In such cases, internet access is provided through designated nodes that have access to external resources.

Traffic is then routed through pre-configured Egress gateways (EgressGateway) according to the specified policy (EgressGatewayPolicy).

Egress gateway node groups

The EgressGateway resource describes a node group that functions as an Egress gateway. To add a node to this group, assign a corresponding label to it:

```
d8 k label node <node name> dedicated/egress=
```

PrimaryIPFromEgressGatewayNodeInterface mode

The primary IP address bound to the public network interface of the node will be used as the IP address. If the active node fails and a new node is assigned, the sender IP address in network packets will change.

To create an Egress gateway, apply the following EgressGateway resource:

```
d8 k apply -f - <<EOF
apiVersion: network.deckhouse.io/v1alpha1
kind: EgressGateway
metadata:
  name: egress-gw
spec:
  nodeSelector:
    dedicated/egress: ""
  sourceIP:

# The primary IP address bound to the public network interface of the node will be
used as the IP address
    mode: PrimaryIPFromEgressGatewayNodeInterface
    primaryIPFromEgressGatewayNodeInterface:

# Since the "public" interface must have the same name on all nodes in the group (for
example, eth1),
      # make sure to configure the network subsystem first on all Egress nodes
      interfaceName: eth1
EOF
```

VirtualIPAddress mode

Alternatively, you can assign a virtual IP address to the node group.

This virtual address will be bound to the master node, providing traffic routing to external services. If the master node fails, all active connections will be terminated, and a new master node will be selected and assigned the same virtual address. The client IP address will remain unchanged for external services.

To create an Egress gateway with a virtual address, apply the following EgressGateway resource:

```
d8 k apply -f - <<EOF
apiVersion: network.deckhouse.io/v1alpha1
kind: EgressGateway
metadata:
  name: egress-gw
spec:
  nodeSelector:
    dedicated/egress: ""
  sourceIP:

# The primary IP address bound to the public network interface of the node will be
used as the IP address
    mode: VirtualIPAddress
    virtualIPAddress:
      # All routes required to access external public services must be configured on
every node
      # The "public" interface must be prepared to automatically configure the
"virtual" IP address as a secondary IP address
      ip: 172.18.18.242
EOF
```

Egress gateway node suitability

The nodes specified by `spec.nodeSelector` of the EgressGateway resource are evaluated and only the suitable ones are selected. A node is considered suitable if it meets the following conditions:

1. The node’s status is Ready.
2. The node isn’t under maintenance, meaning it isn’t cordoned.
3. cilium-agent on the node is Ready.

To check whether a node is suitable for inclusion to the Egress gateway group, run the following commands:

```
# Display nodes specified by spec.nodeSelector:
d8 k get nodes -l dedicated/egress="" -ojson | jq -r '.items[].metadata.name'

# Display nodes in the Ready status:
d8 k get nodes -ojson | jq -r '.items[] | select(.status.conditions[] | select(.type
== "Ready" and .status == "True")) | .metadata.name'

# Display nodes that aren't under maintenance:
d8 k get nodes --field-selector spec.unschedulable=false -ojson | jq -r .items[].metad
ata.name

# Display nodes with cilium-agent running:
d8 k get pods -n d8-cni-cilium -l app=agent -ojson | jq -r '.items[].spec.nodeName'
```

In a node group, one node is designated as the master node. External traffic is routed through this node. The other nodes remain in hot-standby mode. If the active node fails, the active connections will be terminated, and a new master node will be selected from the remaining nodes. Traffic will then be routed through this new node.

Different Egress gateways can use common nodes for operation. Master nodes are selected independently, helping to distribute the load across them.

Traffic redirection policy

The EgressGatewayPolicy resource describes the policy for redirecting application traffic from virtual machines to the specified Egress gateway. The policy and virtual machines are matched by labels.

To add a virtual machine to the policy, assign a label to it:

```
d8 k label vm <virtual machine name> app=backend
```

To set a traffic redirection policy, create an EgressGatewayPolicy resource by specifying criteria in the `.spec.selectors` field to select the virtual machines to which the policy will apply. Use the following example:

```
d8 k apply -f - <<EOF
apiVersion: network.deckhouse.io/v1alpha1
kind: EgressGatewayPolicy
metadata:
  name: egress-gw-policy
spec:
  destinationCIDRs:
    - 0.0.0.0/0
  egressGatewayName: egress-gw
  selectors:
    - podSelector:
        matchLabels:
          # This policy will be applied to all pods with the label app=backend in the
default namespace on all virtual machines
          app: backend
          io.kubernetes.pod.namespace: default
EOF
```

To ensure the EgressGatewayPolicy policy has been properly applied to the virtual machine, run the following command on it to verify network connectivity and routing:

```
curl ifconfig.me
```

Upon verification, you will see the master node's IP address if the Egress gateway is in `PrimaryIPFromEgressGatewayNodeInterface` mode, or the virtual IP address if the Egress gateway is in `VirtualIPAddress` mode.

Routing management

> ⚠ This feature is available only in Enterprise Edition.

To control static routes and IP rules on cluster nodes, use the static-routing-manager module.

To enable the module with default settings, apply the following `ModuleConfig` resource:

```
d8 k apply -f - <<EOF
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: static-routing-manager
spec:
  version: 1
  enabled: true
EOF
```

Routing table

The `RoutingTable` resource describes the target routing table and its associated routes.

To create a route in the main routing table, do the following:

1. Apply the `RoutingTable` resource to create a new route (`10.0.0.0/8 via 192.168.0.1`) on nodes that match the specified nodeSelector:

   ```
   d8 k apply -f - <<EOF
   apiVersion: network.deckhouse.io/v1alpha1
   kind: RoutingTable
   metadata:
   name: myrt-main
   spec:
     ipRoutingTableID: 254 # Main routing table ID is 254
     routes:
     - destination: 10.0.0.0/8
       gateway: 192.168.0.1
     nodeSelector:
       node-role.deckhouse.io: load-balancer
   EOF
   ```

2. To check the new route created in the main routing table, run the following command:

   ```
   ip -4 route ls
   ```

   In the output, you will see a list of routes, including the newly created `10.0.0.0/8 via 192.168.0.1` :

   ```
   ...
   10.0.0.0/8 via 192.168.0.1 dev eth0 realm 216
   ...
   # The routed instruction 'realm 216' is used as a marker to identify the route
   managed by the module (d8 hex = 216 dec)
   ```

To create a route in an additional table, do the following:

1. Apply the `RoutingTable` resource to create a new route (`0.0.0.0/0 via 192.168.0.1`) in table 10000 on nodes that match the specified nodeSelector:

```
d8 k apply -f - <<EOF
apiVersion: network.deckhouse.io/v1alpha1
kind: RoutingTable
metadata:
name: myrt-extra
spec:
  routes:
    - destination: 0.0.0.0/0
      gateway: 192.168.0.1
  nodeSelector:
    node-role.deckhouse.io: load-balancer
status:
  ipRoutingTableID: 10000
# If spec.ipRoutingTableID isn't specified, it will be generated and placed into
status automatically
    ...
EOF
```

2. To check the new route created in the additional table, run the following command:

   ```
   ip -4 route ls table 10000
   ```

   In the output, you will see a list of routes from table 10000, including the newly created `default via 192.168.0.1`:

   ```
   ...
   default via 192.168.0.1 dev eth0 realm 216
   ...
   ```

Routing rules

The `IPRuleSet` resource describes a set of IP rules that will be created on the nodes with the corresponding labels.

To apply a rule, do the following:

1. Create the IPRuleSet resource to create an IP rule on nodes that match the specified nodeSelector:

```
d8 k apply -f - <<EOF
apiVersion: network.deckhouse.io/v1alpha1
kind: IPRuleSet
metadata:
  name: myiprule
spec:
  rules:
    - selectors:
        from:
          - 192.168.111.0/24
          - 192.168.222.0/24
        to:
          - 8.8.8.8/32
          - 172.16.8.0/21
        sportRange:
          start: 100
          end: 200
        dportRange:
          start: 300
          end: 400
        ipProto: 6
      actions:
        lookup:
      routingTableName: myrt-extra
      priority: 50
  nodeSelector:
    node-role.deckhouse.io: load-balancer
EOF
```

2. To ensure the newly created rule was applied, run the following command:

   ```
   ip rule list
   ```

   In the output, you will see a list of configured rules:

   ```
   ...
   50: from 192.168.111.0/24 to 172.16.8.0/21 ipproto tcp sport 100-200 dport
   300-400 lookup 10000 realms 216
   50: from 192.168.222.0/24 to 8.8.8.8 ipproto tcp sport 100-200 dport 300-400
   lookup 10000 realms 216
   50: from 192.168.222.0/24 to 172.16.8.0/21 ipproto tcp sport 100-200 dport
   300-400 lookup 10000 realms 216
   50: from 192.168.111.0/24 to 8.8.8.8 ipproto tcp sport 100-200 dport 300-400
   lookup 10000 realms 216
   ...
   ```

DNS management

To install CoreDNS components for DNS management, use the kube-dns module.

> kube-dns deletes resources previously installed via kubeadm, including Deployment, ConfigMap, and RBAC for CoreDNS.

To enable kube-dns with default settings, use the following ModuleConfig resource:

```
d8 k apply -f - <<EOF
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: kube-dns
spec:
  enabled: true
EOF
```

For detailed description of kube-dns settings, refer to the corresponding article.

DNS configuration example

Example of the kube-dns module configuration using the ModuleConfig resource:

```
d8 k apply -f - <<EOF
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: kube-dns
spec:
  version: 1
  enabled: true
  settings:
    # A list of IP addresses for recursive DNS servers that CoreDNS uses to resolve
external domains
    # By default, it uses the list from /etc/resolv.conf
    upstreamNameservers:
      - 8.8.8.8
      - 8.8.4.4
    # A static list of hosts formatted similarly to /etc/hosts
    hosts:
      - domain: one.example.com
        ip: 192.168.0.1
      - domain: two.another.example.com
        ip: 10.10.0.128
    # A list of additional zones managed by CoreDNS
    stubZones:
      - zone: consul.local
        upstreamNameservers:
          - 10.150.0.1
    # A list of alternative cluster domains resolved along with
global.discovery.clusterDomain
    clusterDomainAliases:
      - foo.bar
      - baz.qux
EOF
```

For detailed description of kube-dns settings, refer to the corresponding article.

Replace a cluster domain

To replace a cluster domain with minimal downtime, follow these steps:

1. Edit the settings of the control-plane manager module responsible for Deckhouse configuration.

   Make changes using the following template:

```
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: control-plane-manager
spec:
  version: 1
  enabled: true
  settings:
    apiserver:
      # A list of SANs certificate options for generating the API server
certificate
      certSANs:
       - kubernetes.default.svc.<old clusterDomain>
       - kubernetes.default.svc.<new clusterDomain>
      serviceAccount:
        # A list of API audiences to include when creating ServiceAccount tokens
        additionalAPIAudiences:
        - https://kubernetes.default.svc.<old clusterDomain>
        - https://kubernetes.default.svc.<new clusterDomain>
        # A list of additional ServiceAccount API token issuers to add as they
are created
        additionalAPIIssuers:
        - https://kubernetes.default.svc.<old clusterDomain>
        - https://kubernetes.default.svc.<new clusterDomain>
```

2. Add a list of alternative cluster domains to the kube-dns module configuration:

```
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: kube-dns
spec:
  version: 1
  enabled: true
  settings:
    clusterDomainAliases:
      - <old clusterDomain>
      - <new clusterDomain>
```

3. Wait for `kube-apiserver` to restart.

4. Replace `clusterDomain` with a new domain in `dhctl config edit cluster-configuration`.

NTP management

Configuring node time synchronization

To configure time synchronization on nodes, use the chrony module or replace it with a custom NTP daemon.

To enable the kube-dns module with default settings, apply the `ModuleConfig` resource, specifying your NTP servers for synchronization. Example of configuration with a default NTP server:

```
d8 k apply -f - <<EOF
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: chrony
spec:
  enabled: true
  settings:
    ntpServers:
      - pool.ntp.org
  version: 1
EOF
```

Using NTP daemons

1. To disable chrony and use custom NTP daemons on nodes, disable the module running the following command:

   ```
   d8 k -ti -n d8-system exec svc/deckhouse-leader -c deckhouse -- deckhouse-
   controller module disable chrony
   ```

   If the command succeeds, you should see a message confirming that the module has been disabled:

   ```
   Module chrony disabled
   ```

2. To enable NTP daemons on nodes, create NodeGroupConfiguration. Below is an example configuration using systemd-timesyncd:

```
d8 k apply -f - <<EOF
apiVersion: deckhouse.io/v1alpha1
kind: NodeGroupConfiguration
metadata:
  name: enable-ntp-on-node.sh
spec:
  weight: 100
  nodeGroups: ["*"]
  bundles: ["*"]
  content: |
    systemctl enable systemd-timesyncd
    systemctl start systemd-timesyncd
EOF
```

Monitoring

Node monitoring

Monitoring

For node groups (NodeGroup resource), DKP exports availability metrics for the group.

What information does Prometheus collect?

All node group metrics have the prefix `d8_node_group_` in their name, and a label with the node group's name `node_group_name`.

The following metrics are collected for each node group:

- `d8_node_group_ready` — the number of nodes in the group that are in `Ready` status;
- `d8_node_group_nodes` — the total number of nodes in the group (in any status);
- `d8_node_group_instances` — the total number of instances in the group (in any status);
- `d8_node_group_desired` — the desired (target) number of `Machines` objects in the group;
- `d8_node_group_min` — the minimum number of instances in the group;
- `d8_node_group_max` — the maximum number of instances in the group;
- `d8_node_group_up_to_date` — the number of nodes in the group in up-to-date state;
- `d8_node_group_standby` — the number of standby nodes (see the `standby` parameter in the group);
- `d8_node_group_has_errors` — one if there are any errors in the node group.

Access control

Role model

Description

DVP provides a standard set of roles for managing access to project and cluster resources. There are two types of roles:

- Use roles: Assigned to project users, allowing them to manage resources **within a specified project**.
- Manage roles: Intended for DVP administrators, granting them permissions to manage resources at the platform-wide level.

Platform permissions are configured using the standard Kubernetes RBAC approach. This involves creating `RoleBinding` or `ClusterRoleBinding` resources to assign the appropriate role.

Use roles

Use roles grant permissions to a user **within a specified project** and define access to project resources. These roles can only be used with a `RoleBinding` resource.

DVP provides the following use roles:

- `d8:use:role:viewer`: Allows viewing project resources and authenticate to the cluster.
- `d8:use:role:user`: Includes all permissions from the `d8:use:role:viewer` role and also allows viewing RBAC secrets and resources, connect to virtual machines, and run the `d8 k proxy` command.
- `d8:use:role:manager`: Includes all permissions from the `d8:use:role:user` role and also allows managing project resources.
- `d8:use:role:admin`: Includes all permissions from the `d8:use:role:manager` role and also allows managing the following resources: `ResourceQuota`, `ServiceAccount`, `Role`, `RoleBinding`, `NetworkPolicy`, and `VirtualImage`.

Example of administrator permissions granted to the `joe` user in the `vms` project:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: project-admin-joe
  namespace: vms
subjects:
- kind: User
  name: joe@example.com # For users.deckhouse.io, the parameter is .spec.email
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: d8:use:role:admin
  apiGroup: rbac.authorization.k8s.io
```

Example of administrator permissions granted to the `vms-admins` user group in the `vms` project:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: project-admin-joe
  namespace: vms
subjects:
- kind: Group
  name: vms-admins # For groups.deckhouse.io, the parameter is .spec.name
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: d8:use:role:admin
  apiGroup: rbac.authorization.k8s.io
```

Manage roles

Manage roles grant permissions to manage the following:

- DVP cluster resources
- DVP module settings
- Module components in projects with the `d8-*` and `kube-*` prefixes

DVP provides the following manage roles, allowing to manage all subsystems of the `all` cluster:

- `d8:manage:all:viewer` : Grants permissions to view module configurations (`moduleConfig` resources) and access cluster-wide resources of these modules.
- `d8:manage:all:manager` : Includes all permissions from the `viewer` role and also allows managing module configurations and cluster-wide resources of these modules.

Example of cluster administrator permissions granted to the `jane` user:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: cluster-admin-jane
subjects:
- kind: User
  name: jane.doe@example.com # For users.deckhouse.io, the parameter is .spec.email
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: d8:manage:all:admin # Manage role name
  apiGroup: rbac.authorization.k8s.io
```

DVP can grant restricted permissions to administrators for managing resources and modules associated with specific subsystems.

To assign network subsystem administrator permissions to the `jane` user, use the following configuration:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: network-admin-jane
subjects:
- kind: User
  name: jane.doe@example.com # For users.deckhouse.io, the parameter is .spec.email
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: d8:manage:networking:admin # Manage role name
  apiGroup: rbac.authorization.k8s.io
```

Subsystem management role names follow the `d8:manage:<SUBSYSTEM>:<ACCESS_LEVEL>` format, where:

- `<SUBSYSTEM>` indicates the subsystem name.
- `<ACCESS_LEVEL>` indicates the access level, similar to the roles for the `all` subsystem.

The subsystems available for manage roles are listed in the following table:

{% include rbac/rbac-subsystems-list.liquid %}

User management

Description

DVP supports both internal user and group management, as well as integration with external authentication providers and protocols, such as:

- GitHub
- GitLab
- Crowd
- Bitbucket Cloud
- LDAP
- OIDC

You can connect multiple external authentication providers at the same time.

Users can access DVP web interfaces, such as Grafana and Console, and use command-line utilities like `d8` or `kubectl` to interact with the DVP APIs, considering the granted access permissions.

For details on granting permissions to users and groups, refer to Role Model.

Create a user

To create a static user, use the User resource.

Before creating a user, generate a password hash using the following command:

```
# To avoid saving the password in the command history, begin the command line with a
space character
# Replace example_password with your password
 echo -n 'example_password' | htpasswd -BinC 10 "" | cut -d: -f2 | tr -d '\n' | base64
-w0; echo
```

Alternatively, you can use Bcrypt.

Example of a manifest for creating a user:

```
apiVersion: deckhouse.io/v1
kind: User
metadata:
  name: joe
spec:
  email: joe@example.com # Used in RoleBinding and ClusterRoleBinding to assign user
permissions
  password: 'JDJ5JDEwJG5qNFZUWW9vVHBQZUsxV1ZaNWtOcnVzTXhDb3ZHcWNFLnhxSHhoMUM0aG9zVVJub
UJkZjJ5'
  ttl: 24h # (Optional) Sets the lifetime of the user account
```

Create a user group

To create a user group, use the Group resource.

Example of a manifest for creating a user group:

```
apiVersion: deckhouse.io/v1alpha1
kind: Group
metadata:
  name: vms-admins
spec:
  # A list of users
  members:
  - kind: User
    name: joe
  name: vms-admins # Used in RoleBinding and ClusterRoleBinding to assign user group
permissions
```

Create a configuration file for remote access

To control the cluster remotely using command-line utilities like `d8` or `kubectl`, create a configuration file:

1. In the ModuleConfig resource of the `user-authn` module, enable access to the DVP API by setting the `.spec.settings.publishAPI.enabled` parameter to `true`.
2. Using the kubeconfigurator web interface, generate a `kubeconfig` file for remote access to the cluster. The `kubeconfig` name is reserved for accessing the web interface that generates the `kubeconfig` file. The access URL is determined by the value of the `publicDomainTemplate` parameter.

   To find out the address for accessing the service, run the following command:

   ```
   d8 k get ingress -n d8-user-authn
   # NAME                    CLASS   HOSTS
   ADDRESS                           PORTS     AGE
   # ...
   # kubeconfig-generator   nginx   kubeconfig.example.com
   172.25.0.2,172.25.0.3,172.25.0.4   80, 443   267d
   # ...
   ```
3. Go to the provided address and log in using the email and password you specified when creating a user.

Configuration of external providers

To configure an external provider, use the DexProvider resource.

GitHub

Example of a manifest for configuring a provider to integrate with GitHub:

```
apiVersion: deckhouse.io/v1
kind: DexProvider
metadata:
  name: github
spec:
  type: Github
  displayName: My Company GitHub
  github:
    clientID: plainstring
    clientSecret: plainstring
```

In a GitHub organization, create a new application:

1. Go to **Settings** → **Developer settings** → **OAuth Apps** → **Register a new OAuth application**.
2. In the **Authorization callback URL** field, enter:
   `https://dex.<modules.publicDomainTemplate>/callback`.

Specify `Client ID` and `Client Secret` that you receive in the custom `DexProvider` resource.

If the GitHub organization is managed by a client, do the following:

1. Go to **Settings** -> **Applications** -> **Authorized OAuth Apps**.
2. Find the created application using its name and click **Send Request** to confirm.
3. Ask the client to confirm the request sent to their email.

Once you go through these steps, your application will be ready for use as an authentication provider via GitHub.

GitLab

Example of a manifest for configuring a provider to integrate with GitLab:

```
apiVersion: deckhouse.io/v1
kind: DexProvider
metadata:
  name: gitlab
spec:
  type: Gitlab
  displayName: Dedicated Gitlab
  gitlab:
    baseURL: https://gitlab.example.com
    clientID: plainstring
    clientSecret: plainstring
    groups:
    - administrators
    - users
```

> `groups` in the above example is a list of allowed GitLab group filters specified by their paths and not by names. The user token will contain a set intersection of GitLab groups and groups from this list. If the set is empty, the authorization will be considered unsuccessful. The user token will contain all GitLab groups if the parameter is not set.

To create an application in GitLab, follow the steps below.

For a self-managed GitLab instance:

1. Go to **Admin area** → **Application** → **New application**.
2. In the **Redirect URI (Callback URL)** field, enter the address: `https://dex.<modules.publicDomainTemplate>/callback`.
3. Select the following scopes:
    - `read_user`
    - `openid`

For a GitLab SaaS instance:

1. Using the Owner or Maintainer role account, go to **User Settings** → **Applications** → **New application**.
2. In the Redirect URI (Callback URL) field, enter the address: `https://dex.<modules.publicDomainTemplate>/callback`.
3. Select the following scopes:
    - `read_user`
    - `openid`

For GitLab 16.0 or newer:

1. When creating the application, mark it as **trusted**. Trusted applications are automatically authorized on GitLab OAuth flow.
2. Use `Application ID` and `Secret` that you receive in the custom `DexProvider` resource.

Atlassian Crowd

Example of a manifest for configuring a provider to integrate with Atlassian Crowd:

```
apiVersion: deckhouse.io/v1
kind: DexProvider
metadata:
  name: crowd
spec:
  type: Crowd
  displayName: Crowd
  crowd:
    baseURL: https://crowd.example.com/crowd
    clientID: plainstring
    clientSecret: plainstring
    enableBasicAuth: true
    groups:
    - administrators
    - users
```

To create a generic application in Atlassian Crowd, follow these steps:

1. Go to **Applications → Add application**.
2. Use `Application Name` and `Password` that you receive in the DexProvider resource.
3. Specify CROWD groups in lowercase for the `DexProvider` resource.

Bitbucket Cloud

Example of a manifest for configuring a provider to integrate with Bitbucket:

```
apiVersion: deckhouse.io/v1
kind: DexProvider
metadata:
  name: bitbucket
spec:
  type: BitbucketCloud
  displayName: Bitbucket
  bitbucketCloud:
    clientID: plainstring
    clientSecret: plainstring
    includeTeamGroups: true
    teams:
    - administrators
    - users
```

To set up authentication in Bitbucket, follow these steps:

1. In the workspace menu, create a new OAuth consumer.

2. Go to **Settings** → **OAuth consumers** → **New application** and set the following address in the **Callback URL** field: `https://dex.<modules.publicDomainTemplate>/callback`.
3. Allow access for `Account: Read` and `Workspace membership: Read`.
4. Specify `Key` and `Secret` that you receive in the custom `DexProvider` resource.

LDAP

Example of a manifest for configuring a provider to integrate with Active Directory:

```
apiVersion: deckhouse.io/v1
kind: DexProvider
metadata:
  name: active-directory
spec:
  type: LDAP
  displayName: Active Directory
  ldap:
    host: ad.example.com:636
    insecureSkipVerify: true

    bindDN: cn=Administrator,cn=users,dc=example,dc=com
    bindPW: admin0!

    usernamePrompt: Email Address

    userSearch:
      baseDN: cn=Users,dc=example,dc=com
      filter: "(objectClass=person)"
      username: userPrincipalName
      idAttr: DN
      emailAttr: userPrincipalName
      nameAttr: cn

    groupSearch:
      baseDN: cn=Users,dc=example,dc=com
      filter: "(objectClass=group)"
      userMatchers:
      - userAttr: DN
        groupAttr: member
      nameAttr: cn
```

To set up authentication in LDAP, follow these steps:

1. Create a read-only user (service account) in LDAP.
2. Specify the user path and password you receive in the `bindDN` and `bindPW` parameters of the custom `DexProvider` resource.
3. If LDAP has anonymous read access configured, these settings can be skipped.
4. In the `bindPW` parameter, specify the password in plain text. Hashed password can’t be used.

OIDC (OpenID Connect)

Authentication via an OIDC provider requires registering a client or creating an application. To do this, follow a guide from a respective provider: Okta, Keycloak, Gluu, or Blitz.

Specify `clientID` and `clientSecret` that you receive in the custom `DexProvider` resource.

Below are several manifests with configuration examples.

Okta

Example of a manifest for configuring a provider to integrate with Okta:

```
apiVersion: deckhouse.io/v1
kind: DexProvider
metadata:
  name: okta
spec:
  type: OIDC
  displayName: My Company Okta
  oidc:
    issuer: https://my-company.okta.com
    clientID: plainstring
    clientSecret: plainstring
    insecureSkipEmailVerified: true
    getUserInfo: true
```

Blitz Identity Provider

When registering an application with Blitz Identity Provider, specify the URL to redirect users after authorization. When using `DexProvider`, specify `https://dex.<publicDomainTemplate>/`, where `publicDomainTemplate` is the cluster DNS name template configured in the `global` module.

Example of a manifest for configuring a provider to integrate with Blitz Identity Provider:

```
apiVersion: deckhouse.io/v1
kind: DexProvider
metadata:
  name: blitz
spec:
  displayName: Blitz Identity Provider
  oidc:
    basicAuthUnsupported: false
    claimMapping:
      email: email
      groups: your_claim
# Claim to receive user groups. User groups are configured on the Blitz Identity
Provider's side
    clientID: clientID
    clientSecret: clientSecret
    getUserInfo: true
    insecureSkipEmailVerified: true # If email verification isn't required, set to
true
    insecureSkipVerify: false
    issuer: https://yourdomain.idblitz.com/blitz
    promptType: consent
    scopes:
    - profile
    - openid
    userIDKey: sub
    userNameKey: email
  type: OIDC
```

To ensure proper logout from applications, involving token callbacks and requiring re-authorization, set the `promptType` parameter to `login`.

To ensure detailed user access to applications, do the following:

1. Add the `allowedUserGroups` parameter to ModuleConfig of the respective application.
2. Assign groups to the user. Group names must match those configured in Blitz and Deckhouse.

Example of adding groups for the Prometheus module:

```
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: prometheus
spec:
  version: 2
  settings:
    auth:
      allowedUserGroups:
        - adm-grafana-access
        - grafana-access
```

Projects

Description

DVP projects (the Project resource) provide isolated environments for creating user resources.

Project settings let you set resource quotas and restrict network communication both within and outside DVP.

You can create a project using a template (the ProjectTemplate resource).

> ⚠ If you modify a project template, all projects created from it will be updated to match the modified template.

Default project templates

DVP provides the following set of project templates:

- `empty` : A blank template without predefined resources.
- `default` : A template for main project use cases:
  - Resource restrictions
  - Network isolation
  - Automated alerts and logging
  - Security profile selection
  - Project administration setup
- `secure` : Includes all features of the `default` template and some additional features:
  - Audit rules for project Linux users accessing the kernel

To get a list of all available project template parameters, run the following command:

```
d8 k get projecttemplates <PROJECT_TEMPLATE_NAME> -o jsonpath='{.spec.parametersSchema
.openAPIV3Schema}' | jq
```

Create a project

1. For a new project, create a Project resource, specifying the project template name in the `.spec.projectTemplateName` field.
2. In the `.spec.parameters` parameter of the Project resource.

   Example of creating a project using the Project resource based on the `default` project template:

   ```
   apiVersion: deckhouse.io/v1alpha2
   kind: Project
   metadata:
     name: my-project
   spec:
     description: This is an example from the Deckhouse documentation.
     projectTemplateName: default
     parameters:
       resourceQuota:
         requests:
           cpu: 5
           memory: 5Gi
           storage: 1Gi
         limits:
           cpu: 5
           memory: 5Gi
       networkPolicy: Isolated
       podSecurityProfile: Restricted
       extendedMonitoringEnabled: true
       administrators:
       - subject: Group
         name: k8s-admins
   ```

3. To check the project status, run the following command:

   ```
   d8 k get projects my-project
   ```

   A properly created project has the `Deployed` (synchronized) status. If the displayed status is `Error` and you want to see error details, add the `-o yaml` argument to the command:

   ```
   d8 k get projects my-project -o yaml
   ```

Create a custom project template

The default project templates cover main project use cases and demonstrate template capabilities.

To create a project template of your own, follow these steps:

1. Select one of the default templates, such as `default`.

2. Make a copy to a separate file (for example, to `my-project-template.yaml`) using the following command:

```
d8 k get projecttemplates default -o yaml > my-project-template.yaml
```

3. Open and edit `my-project-template.yaml` to customize the template.

> Make sure to modify not only the template, but also the corresponding parameter schema.
>
> Project templates support all Helm template functions.

4. Change template name in the `.metadata.name` field.
5. Apply the customized template using the following command:

```
d8 k apply -f my-project-template.yaml
```

6. To ensure the modified template is available, run the following command:

```
d8 k get projecttemplates <NEW_TEMPLATE_NAME>
```

Platform update

Platform update

Automatic Update

The platform supports automatic updates, allowing the system to stay up-to-date without user intervention. This mode is recommended for most users as it minimizes the risk of system obsolescence and ensures timely access to functional and security improvements.

Update Windows

To provide flexibility in the update process, you can set update windows. These are specific time intervals during which the system will be updated. This configuration minimizes the impact of updates on workflows and allows them to be aligned with the organization’s schedule.

Manual Update

Manual updating is intended for users who prefer full control over the update process. This approach requires the user to manage all stages, from downloading updates to installing them. Manual updating allows for a thorough review of changes before their application.

Manual update mode

Coming soon…

Removing the platform

To delete a cluster, several steps need to be followed:

1. Remove all additional nodes from the cluster:
    1. Remove the node from the Kubernetes cluster:

        ```
        d8 k drain <node> --ignore-daemonsets --delete-emptydir-data
        d8 k delete node <node>
        ```

    2. Run the cleanup script on the node:

        ```
        bash /var/lib/bashible/cleanup_static_node.sh --yes-i-am-sane-and-i-understand-what-i-am-doing
        ```

2. Check the release channel set in the cluster. To do this, run the command:

    ```
    d8 k get mc deckhouse -o jsonpath='{.spec.settings.releaseChannel}'
    ```

3. Run the Deckhouse installer:

    ```
    docker run --pull=always -it [<MOUNT_OPTIONS>] \
       registry.deckhouse.io/deckhouse/<DECKHOUSE_REVISION>/
    install:<RELEASE_CHANNEL> bash
    ```

    where:

    - `<MOUNT_OPTIONS>` — the options for mounting files into the installer container, such as SSH access keys;
    - `<DECKHOUSE_REVISION>` — edition of the platform (e.g., `ee` — for Enterprise Edition, `ce` — for Community Edition, etc.)
    - `<RELEASE_CHANNEL>` — release channel of the platform in kebab-case. It should match the one set in `config.yaml` during the installation:
        - `alpha` — for the *Alpha* release channel;
        - `beta` — for the *Beta* release channel;
        - `early-access` — for the *Early Access* release channel;
        - `stable` — for the *Stable* release channel;
        - `rock-solid` — for the *Rock Solid* release channel.

    Example of running the installer container for the CE edition:

```
docker run -it --pull=always \
   -v "$PWD/dhctl-tmp:/tmp/dhctl" \
   -v "$HOME/.ssh/:/tmp/.ssh/" registry.deckhouse.io/deckhouse/ce/install:stable
bash
```

4. Execute the cluster removal command:

   ```
   dhctl destroy --ssh-user=<USER> \
      --ssh-agent-private-keys=/tmp/.ssh/id_rsa \
      --yes-i-am-sane-and-i-understand-what-i-am-doing \
      --ssh-host=<MASTER_IP>
   ```

   where:

   - `<USER>` — the user of the remote machine from which the installation was performed. If `<USER>` needs to perform actions via `sudo`, add the `--ask-become-pass` flag;
   - `<MASTER_IP>` — IP address of the master node in the cluster.

   The installer will connect to the master node and remove all Deckhouse components and the Kubernetes cluster from it.

Stronghold configuration

Coming soon…

Administration

Deckhouse Stronghold administrator guide

This section is intended for administrators of Deckhouse Stronghold.

The platform's Administrator Guide includes the following sections:

- Running on Linux OS
  - Running on Linux OS – a quick start with an example of configuring a high-availability cluster.
  - Configuration – a guide to the Standalone execution configuration parameters.
- Running on Deckhouse Kubernetes Platform
  - Platform Installation - environment preparation, installation, and initial access setup.
  - Platform Configuration – cluster node management, networking, storage systems, virtualization, and access control.
  - Platform Update – configuring update modes and windows, and manual approval of updates.
  - Platform Removal – the process of removing the platform.
- Audit
  - Introduction - what Stronghold audit logs contain, which backends are supported, and how to configure auditing safely.
  - Audit log record schema - audit record structure, key objects, and protection of sensitive data.
  - Audit log filtering - select audit records by condition and configure a fallback device.
  - Audit field exclusion - remove selected fields from audit records before they are stored.
- Backups
  - Introduction - overview of manual and automated backups for Stronghold integrated storage.
  - Save a storage snapshot - create a snapshot manually through CLI or API.
  - Inspect a snapshot - locally inspect snapshot contents and basic consistency before restore.
  - Restore from a snapshot - restore a Stronghold cluster from a saved snapshot.
  - Automated snapshots - configure schedules, storage targets, and status checks for automated backups.
- KV replication
  - Introduction - pull-based KV1/KV2 replication between Stronghold clusters. English documentation is in development.
- Namespaces
  - Introduction - isolate configuration and secrets between namespaces, manage them through CLI and API, and use Namespace API Lock.

- Cryptographic algorithms
  - Introduction - overview of TLS, storage encryption, HSM, and the algorithms available in PKI and Transit.
- Plugins
  - Introduction - overview of built-in and external Stronghold plugins and the differences between Standalone and DKP.
  - Plugins in Standalone - plugin directory, registration, versioning, and mounting of external plugins on Linux servers.
  - Plugins in DKP - plugin delivery through `ModuleConfig`, registration, and enablement in Deckhouse Kubernetes Platform.
- KMS and HSM
  - HSM support - PKCS11-based HSM integration for auto-unseal and root key protection; currently supported only for Standalone installations.
  - Yandex Cloud KMS - configure `seal "yandexcloudkms"` for auto-unseal and root key protection; currently supported only for Standalone installations.
  - Double encryption - the `seal wrap` mechanism that adds an extra encryption layer for critical data.

If you have any questions, you can ask for assistance in our Telegram channel. We will be happy to help and provide guidance.

If you are using the Enterprise edition, you can also email us at support@deckhouse.io for additional support.

Audit

Audit in Stronghold

Auditing in Stronghold is used to record API requests and responses. Since every operation in Stronghold is performed through the API, audit logs let you reconstruct user, service, and administrator activity and use that data for investigations, monitoring, and compliance controls.

What gets written to the audit log

When an audit device is enabled, Stronghold records:

- API requests;
- responses to those requests;
- errors that occur while processing operations.

As a result, the audit log becomes the primary source of information about who accessed which path, when an operation happened, what kind of operation was performed, and what the result was.

Paths that are not audited

Some system paths bypass auditing:

```
sys/init
sys/seal-status
sys/seal
sys/step-down
sys/unseal
sys/leader
sys/health
sys/rekey/init
sys/rekey/update
sys/rekey/verify
sys/rekey-recovery-key/init
sys/rekey-recovery-key/update
sys/rekey-recovery-key/verify
sys/storage/raft/bootstrap
sys/storage/raft/join
sys/internal/ui/feature-flags
```

Depending on listener settings, the following unauthenticated paths may also bypass auditing:

```
sys/metrics
sys/pprof/*
sys/in-flight-req
```

These exceptions should be considered when designing your overall observability and investigation model.

Format and protection of audit data

Each audit log entry is a JSON object. The `type` field indicates the record type:

- `request` for request entries;
- `response` for response entries.

By default, sensitive string values in audit logs are not stored in plaintext. Instead, Stronghold hashes them using HMAC-SHA256 and a salt specific to the audit device. This reduces the risk of leaking secrets through logs while still allowing value verification through the audit hash mechanism.

Keep in mind:

- not all data types are masked in exactly the same way;
- once an audit device is disabled, its hashing salt is lost, so comparing old values through that device is no longer possible;
- raw logging of sensitive data should only be used deliberately and under tight control.

For a more detailed explanation of audit record contents and storage format, see Audit log record schema.

Why multiple audit devices are recommended

If multiple audit devices are enabled, Stronghold tries to write an event to all of them. This gives you several benefits:

- redundant copies of audit records;
- the ability to send logs to different systems;
- better visibility into delivery issues and possible log tampering.

In practice, this means the complete audit trail should be treated as the union of logs from all configured devices.

> ⚠ It is recommended to use at least two audit devices, or at minimum one primary device plus a well-designed fallback strategy. Audit failures can affect Stronghold's ability to serve requests.

What happens when audit devices fail

Auditing is security-critical in Stronghold, so failure handling depends on the failure mode:

- if at least one audit device writes the event successfully, the request can usually complete successfully;
- if an audit device enters a blocking state and the write never completes, requests to Stronghold may hang until the problem is resolved;
- when using network-based backends, you must separately account for timeouts, socket availability, and possible UDP packet loss.

Because of this, you should ensure that every cluster node can actually write to the configured audit backend before enabling it.

Supported audit backends

Stronghold supports the same basic audit backends as Vault:

- `file` for writing to a file;
- `syslog` for sending logs to the local syslog agent;
- `socket` for sending logs to a TCP, UDP, or UNIX socket.

`file`

This is the most predictable backend for local storage and later forwarding to a centralized logging system.

Example:

```
stronghold audit enable file file_path=/var/log/stronghold_audit.log
```

`syslog`

Useful on Unix systems that already rely on a syslog-based logging stack.

Example:

```
stronghold audit enable syslog tag="stronghold" facility="AUTH"
```

> ⚠ Audit messages can be large, so for `syslog` it is safer to use a reliable transport configuration or enable a `file` backend alongside it.

`socket`

Useful for integration with external systems over TCP, UDP, or UNIX sockets.

Example:

```
stronghold audit enable socket address=127.0.0.1:9090 socket_type=tcp
```

If you use `UDP`, account for the possibility of silent message loss. For production scenarios, it is best to combine `socket` with another, more reliable audit device.

Basic operations

Enable an audit device:

```
stronghold audit enable file file_path=/var/log/stronghold_audit.log
```

List enabled devices:

```
stronghold audit list
```

Disable a device:

```
stronghold audit disable file/
```

Advanced capabilities

Stronghold also supports advanced ways to fine-tune audit logs:

- Audit log record schema explains the audit record structure, the `auth`, `request`, and `response` objects, and protection of sensitive values;
- Audit log filtering lets you send only selected records to a specific device;

- Audit field exclusion lets you remove selected fields from records before they are written.

These capabilities are useful for separating audit streams, reducing log volume, and limiting retention of selected sensitive fields, but they should be used carefully.

> ⚠ Filtering and field exclusion are advanced features. Incorrect configuration can lead to gaps in audit logs or loss of important investigation data.

Recommendations

- Enable auditing immediately after Stronghold initialization.
- Use more than one audit device in production.
- Test filter and exclusion rules in a non-production environment.
- Treat auditing as part of the security model, not just a troubleshooting tool.

Audit log record schema

Stronghold writes audit events as individual JSON objects separated by newlines. Audit log records contain both shared attributes that apply to all API calls and request or response fields that depend on the specific operation.

Audit log records represent:

- API requests received by Stronghold;
- API responses returned by Stronghold.

You can correlate a response record with its matching request by using `request.id`, which appears in both records.

Top-level record structure

Regardless of record type, the following top-level fields are typically used:

Attribute	Type	Description
`auth`	`object`	Authentication object describing the subject that performed the API call.
`error`	`string`	Error text if the operation failed. Usually omitted for successful requests.
`forwarded_from`	`string`	Host and port of the node that forwarded the request. Present only in matching scenarios.
`request`	`object`	Request object describing request details.
`response`	`object`	Response object. Present in `response` records.
`time`	`string`	Event time in ISO 8601 format.
`type`	`string`	Record type: `request` or `response`.

Example request record

```
{
  "type": "request",
  "time": "2025-06-05T16:10:22.292517Z",
  "request": {
    "id": "6e3d8a4a-1c4b-4f7e-8d91-9b0df2c7a111"
  },
  "auth": {
    "client_token": "hmac-sha256:..."
  },
  "error": "..."
}
```

Example response record

```
{
  "type": "response",
  "time": "2025-06-05T16:10:22.292639Z",
  "request": {
    "id": "6e3d8a4a-1c4b-4f7e-8d91-9b0df2c7a111"
  },
  "response": {
    "data": {}
  },
  "auth": {
    "client_token": "hmac-sha256:..."
  }
}
```

The `auth` object

Stronghold includes only relevant authentication attributes in the `auth` object. For example, unauthenticated requests do not include `client_token` or `accessor`, and `metadata` is omitted if the token has no metadata.

In practice, the `auth` object most often contains:

- `accessor`
- `client_token`
- `display_name`
- `token_type`
- `token_issue_time`
- `token_ttl`
- `metadata`
- `entity_id`
- `policies`
- `identity_policies`
- `token_policies`
- `policy_results`

These fields help explain which subject executed the request, under which authentication context, and which policies led to the request being allowed.

The `request` object

The `request` object describes the technical details of the incoming call. The most important fields are:

- `id` for the unique request identifier;
- `operation` for the type of operation: `create`, `read`, `update`, `delete`, `list`;
- `namespace` for namespace ID and path;

- `path` for the API path that handled the request;
- `request_uri` for the original request URI when it differs from `path`;
- `mount_accessor`, `mount_point`, `mount_type` for mount metadata;
- `mount_running_version`, `mount_running_sha256`, `mount_is_external_plugin` for plugin or builtin engine details;
- `remote_address`, `remote_port` for client network information;
- `headers` for request headers, if configured for logging;
- `client_id`, `client_token`, `client_token_accessor` for client and token identifiers;
- `data` for the request payload;
- `wrap_ttl` when response wrapping is requested.

A simplified example:

```
{
  "id": "",
  "operation": "",
  "namespace": {
    "id": "",
    "path": ""
  },
  "path": "",
  "mount_point": "",
  "mount_type": "",
  "remote_address": "",
  "remote_port": 1234,
  "headers": {},
  "data": {}
}
```

The `response` object

The `response` object describes the result of the API request. It can contain:

- `auth` when the operation returns a token;
- `headers` for response headers;
- `redirect` for auth backend redirects;
- `warnings` for API warnings;
- `data` for the response payload;
- `secret` for leased secret metadata;
- `wrap_info` for wrapping token properties;
- `mount_class`, `mount_accessor`, `mount_point`, `mount_type` for mount details;
- `mount_running_plugin_version`, `mount_running_sha256`, `mount_is_external_plugin` for plugin details.

A simplified example:

```
{
  "data": {},
  "headers": {},
  "mount_point": "",
  "mount_type": "",
  "warnings": [""],
  "secret": {
    "lease_id": ""
  },
  "wrap_info": {
    "token": "",
    "ttl": 60
  }
}
```


Protection of sensitive data

By default, Stronghold does not store most sensitive string values in plaintext. Instead, they are hashed using HMAC-SHA256 and a salt specific to the audit device.

This means:

- secrets are not written to the log in plaintext;
- values can still be verified later through the audit hash mechanism;
- once an audit device is disabled, its salt is no longer available, so old values can no longer be matched through that device.

Keep in mind that hashing primarily applies to string values received in JSON or returned in JSON. Other types, such as integers and booleans, may still be stored in plaintext.

Practical notes

- The complete audit trail should be built from the union of all enabled audit devices.
- If `filter` and `exclude` are enabled, a given device stores the already filtered or modified version of the record.
- For most engineering tasks, understanding the top-level structure and the `auth`, `request`, and `response` objects is enough.

Related pages

- Audit in Stronghold
- Audit log filtering
- Audit field exclusion

Audit field exclusion

Stronghold supports enabling audit devices with the `exclude` option, which removes selected fields from audit records before they are written to the log. This makes it possible to fine-tune the contents of records for each device.

> ⚠ Audit field exclusion is an advanced feature. Using exclusions can cause loss of important information in audit logs. Always test the configuration in a non-production environment.

When an audit device is enabled with exclusions, each audit record is checked against optional conditions before it is written. If a condition matches, the specified fields are removed from the record. A single device can have multiple condition/field combinations.

The behavior of existing devices and new devices without exclusions does not change.

The `exclude` option

The value of `exclude` must be a valid JSON array of exclusion objects.

Exclusion object

- `condition (string, optional)` is a predicate expression in bexpr syntax. When it matches, Stronghold removes the fields listed in `fields`. If `condition` is omitted or an empty string, the rule is applied unconditionally.
- `fields (string[], required)` is an array of fields to remove, expressed in JSON Pointer syntax.

```
[
  {
    "condition": "",
    "fields": [ "" ]
  }
]
```

“Source of truth” semantics

Exclusion conditions are always evaluated against the original, unmodified audit record. Removing fields with one rule **does not affect** evaluation of later rules. This guarantees predictable behavior regardless of rule order.

Exclusion examples

Exclude response data

Remove the `data` field from the response in any audit record:

```
[
  {
    "fields": [ "/response/data" ]
  }
]
```

Exclude request data for transit mounts

Remove the `data` field from the request for records where `mount_type == "transit"`:

```
[
  {
    "condition": "\"/request/mount_type\" == transit",
    "fields": [ "/request/data" ]
  }
]
```

Multiple exclusions

Remove `data` from request and response for `transit`, and also remove `entity_id` from `auth` when `client_token` starts with `hmac`:

```
[
  {
    "condition": "\"/request/mount_type\" == transit",
    "fields": [ "/request/data", "/response/data" ]
  },
  {
    "condition": "\"/auth/client_token\" matches \"hmac.+\"",
    "fields": [ "/auth/entity_id" ]
  }
]
```

Condition syntax

Conditions in `condition` support two field reference formats:

1. **Quoted JSON Pointer**. Example: `"/request/mount_type" == transit`
2. **Native bexpr format**. Example: `request.mount_type == "transit"`

Supported operators include:

- `==`
- `!=`
- `matches`
- `in`
- `not in`

- `and`
- `or`
- `not`

> Unlike `filter`, `exclude` conditions are evaluated against the full audit record. All record fields are available, not just a limited set of top-level properties.

Audit record structure

The fields available in exclusion conditions follow the actual JSON structure of a Stronghold audit record.

In practice this means:

- for JSON Pointer references, use the real path of the field in the JSON structure;
- for bexpr conditions, you can use the equivalent dot notation;
- if a field is an object, you can address both the object itself and its nested fields.

Typical examples:

- `/request/data` <-> `request.data`
- `/request/mount_type` <-> `request.mount_type`
- `/request/namespace/id` <-> `request.namespace.id`
- `/request/namespace/path` <-> `request.namespace.path`
- `/request/request_uri` <-> `request.request_uri`
- `/auth/entity_id` <-> `auth.entity_id`
- `/response/data` <-> `response.data`
- `/response/wrap_info/token` <-> `response.wrap_info.token`

If you are unsure whether a path is correct, first verify how the field is serialized in the relevant audit record type (`request` or `response`) and only then add the exclusion rule.

Practical example

Enable a `file` audit device and exclude response data for `kv` mounts:

```
stronghold audit enable              \
  -path filtered-file                \
  file                               \
  file_path=/logs/audit.log          \
  exclude='[{"condition": "\"/request/mount_type\" == kv", "fields": ["/response/
data"]}]'
```

Combine filtering and exclusions:

```
stronghold audit enable                    \
  -path transit-only                       \
  file                                     \
  filter='mount_type == "transit"'         \
  file_path=/logs/transit.log              \
  exclude='[{"fields": ["/request/data"]}]'
```

In this example the device:

1. Accepts only records from `transit` mounts.
2. Removes `request.data` from them before writing the record.

Audit log filtering

Stronghold supports enabling audit devices with the `filter` option, which limits which records are written to a specific audit log. Filters are defined when the device is enabled and cannot be changed afterward. To change a filter, you must disable and re-enable the device.

> ⚠ Audit log filtering is an advanced feature. Using only filtered devices without a configured fallback can lead to gaps in audit logs. Always test the configuration in a non-production environment.

When an audit device is enabled with a filter, each audit record is matched against a predicate expression. Only matching records are written to the log. The behavior of existing devices and new devices without a filter does not change.

Fallback devices

Filtering makes auditing more flexible, but it also creates a risk of losing records. For example, if one device is configured for `operation == "read"` and another for `operation == "write"`, then records with `operation == "update"` will not reach either of them.

A fallback device stores all records that were not accepted by any filtered device. This helps preserve the security model in which all requests and responses should be logged before secret data is returned to the client.

Installations that rely **only** on filtered devices should always configure a fallback device.

> ℹ Only one fallback device can exist in a Stronghold installation.

Limitations

- A filter cannot be added to an already enabled device. It can only be set at enable time.
- Filtering is supported for all audit device types: `file`, `socket`, and `syslog`.
- You cannot specify both `filter` and `fallback=true`. These options are mutually exclusive.

- Only one fallback device is allowed.
- A fallback device can be the only audit backend, in which case it receives all records.

Filtering and test messages

When an audit device is enabled, Stronghold sends a test message. If the filter does not match the properties of that test message, the message will not appear in the device log. The enable operation is still considered successful.

Default test message properties:

Property	Value
`mount_point`	empty string
`mount_type`	empty string
`namespace`	empty string
`operation`	`update`
`path`	`sys/audit/test`

Available properties for filtering

Filters can reference only the following audit record properties:

Property	Example	Description
`mount_point`	`mount_point == "auth/oidc"`	Log records for the `auth/oidc` mount point
`mount_type`	`mount_type == "kv-v2"`	Log records generated by `kv-v2` plugins
`namespace`	`namespace != "admin/"`	Log records not coming from the `admin` namespace
`operation`	`operation == "read"`	Log all read operations
`path`	`path == "auth/approle/login"`	Log activity on the AppRole login path

> Non-root namespace paths must end with `/` for matching to work correctly. The root namespace has no path and matches only the empty string. To filter the root namespace, use `namespace == ""`.

Filters use bexpr syntax and support:

- comparison operators `==` and `!=`;

- the `matches` operator for regular expressions;
- compound expressions with `and`, `or`, and `not`.

Fallback metrics

Stronghold publishes telemetry metrics related to fallback auditing:

- `audit.fallback.success` increments when a fallback device successfully writes an audit event that was not successfully written by any non-fallback device;
- `audit.fallback.miss` increments when an event was not written by any filtered device and no fallback device is configured.

The `audit.fallback.miss` metric is especially useful for detecting audit event loss in configurations that use filtered devices only.

Practical example

Assume you already have an audit file named `stronghold-audit.log` and want to route key/value (`kv`) events to a separate `kv-audit.log` file.

1. Enable a `file` audit device filtered by `mount_type`:

```
stronghold audit enable                \
  -path kv-only                        \
  file                                 \
  filter='mount_type == "kv"'          \
  file_path=/logs/kv-audit.log
```

2. Enable a fallback device:

```
stronghold audit enable                \
  -path=my-fallback                    \
  -description="fallback device"       \
  file                                 \
  fallback=true                        \
  file_path=/tmp/kv-audit.fallback.log
```

3. Verify that the devices are enabled:

```
stronghold audit list --detailed
```

4. Enable the KV secrets engine:

```
stronghold secrets enable -path my-kv kv-v2
```

5. Write a secret:

```
stronghold kv put -mount=my-kv my_secret the_value=always_angry
```

After that, `kv-audit.log` will contain records for operations on `my-kv`, while the fallback device will capture records that do not match the filter.

Backup and Restore

Overview

Stronghold backups are based on snapshots of integrated Raft storage. A snapshot lets you save cluster state and later use it to restore data after a failure or during planned maintenance operations.

Stronghold supports two main scenarios:

- **manual snapshots**, where an administrator explicitly saves a snapshot file and restores it manually if needed;
- **automated snapshots**, where Stronghold saves backups on a schedule to local disk or S3-compatible storage.

When to use snapshots

Snapshots are useful for:

- protection against data corruption or operator mistakes;
- preparation for upgrades and other risky operations;
- disaster recovery;
- storing backups outside the cluster or hosting site.

Important considerations

- Snapshots apply to Stronghold integrated Raft storage.
- Snapshots work only for clusters that use integrated Raft storage as the primary storage backend.
- If Stronghold uses `etcd`, `postgresql`, or another external system as a backend, it is necessary to use the native backup mechanisms of those systems, rather than the built-in Stronghold commands for Raft snapshots — `stronghold operator raft snapshot`.
- Data remains encrypted in the backup.
- To regain access after restore, you need the correct unseal or recovery keys according to your cluster configuration.
- Automated snapshots are an administrative feature and require a deliberate retention policy.

Available pages

- Save a storage snapshot for manual snapshot creation through CLI and API.
- Inspect a snapshot for local validation of snapshot contents and basic consistency without restoring it to a cluster.

- Restore from a snapshot for cluster restoration from an existing snapshot.
- Automated snapshots for schedule, storage, and status management of automated backups.

Recommendations

- Keep backups outside the same environment they are intended to protect.
- Periodically verify snapshot creation, inspect snapshots with `stronghold operator raft snapshot inspect`, and test restore procedures.
- For production environments, prefer external object storage over local disk whenever possible.

Save a storage snapshot

A manual snapshot lets you save the current state of Stronghold integrated storage to a file and use it as a backup.

Before you start

- Make sure Stronghold uses integrated Raft storage.
- Make sure Raft is used as the primary storage backend, not only as `ha_storage`.
- Confirm that you have administrative access to the cluster.
- Prepare a secure location for storing the snapshot file outside the production environment.
- Make sure you can contact the holders of the unseal or recovery keys if restore becomes necessary.

> ⚠ The `stronghold operator raft snapshot save` command applies only to clusters that use integrated Raft storage. If Stronghold uses `etcd`, `postgresql`, or another external backend, back up that storage system with its own native backup mechanism.

Save a snapshot with CLI

To create a snapshot, run:

```
d8 stronghold operator raft snapshot save backup.snap
```

Stronghold writes the snapshot to the local `backup.snap` file.

After creating the backup, it is recommended to run snapshot inspection to confirm that the file can be read and contains the expected data structure.

Save a snapshot with API

To create a snapshot through the API, use the `GET /sys/storage/raft/snapshot` endpoint:

```
curl \
  --request GET \
  --header "X-Vault-Token: ${VAULT_TOKEN}" \
  "${VAULT_ADDR}/v1/sys/storage/raft/snapshot" > backup.snap
```

Practical recommendations

- Store snapshot files in a restricted and secure location.
- After creating a snapshot, run `stronghold operator raft snapshot inspect` or follow Inspect a snapshot as a baseline backup validation step.
- Use automated snapshots for scheduled backups.
- If your environment uses replication, define a separate backup strategy for each cluster domain.

Inspect a snapshot

The `stronghold operator raft snapshot inspect` command lets you inspect an existing snapshot file without restoring it into a cluster.

Unlike `snapshot save` and `snapshot restore`, this command works **locally** and does not require access to a running Stronghold server. That makes it useful for backup validation, initial troubleshooting, and snapshot analysis before a restore.

> ⚠ The `inspect` command applies to integrated Raft storage snapshots. If Stronghold uses `etcd`, `postgresql`, or another external backend, validate backups with the tools and procedures of that storage system.

Basic usage

```
stronghold operator raft snapshot inspect <snapshot_file>
```

Example:

```
stronghold operator raft snapshot inspect raft.snap
```

As described for Vault 1.21, the command prints snapshot metadata and a table that shows the number of keys and the amount of space used by each key group.

What the command shows

The default output includes:

- snapshot metadata such as `ID`, `Size`, `Index`, `Term`, and `Version`;
- grouped storage keys;
- the number of keys in each group;

- the total size of data in each group.

Example abbreviated output:

```
ID           bolt-snapshot
Size         93290
Index        8957
Term         11
Version      1

Key Name                                        Count      Size
--------                                        -----      ----
wal/logs                                        54         9 KB
index/pages                                     14         26.1 KB
sys/policy                                      3          3.4 KB
core/cluster                                    2          236 B
```

Main flags

Flag	Description
`-details`	Enables detailed key analysis and prints the grouped key table. Enabled by default.
`-depth`	Controls grouping depth by path segments.
`-filter`	Limits output to keys matching a specific prefix.
`-format`	Output format: `table` or `json`.
`-validate`	Runs additional snapshot consistency checks.

Common scenarios

Quick validation after backup creation

You can inspect the file immediately after creating it:

```
stronghold operator raft snapshot save /backup/raft.snap
stronghold operator raft snapshot inspect /backup/raft.snap
```

Validate snapshot consistency

Use the `-validate` flag for a stronger backup check:

```
stronghold operator raft snapshot inspect -validate raft.snap
```

This helps confirm that:

- the snapshot is not empty;
- `state.bin` can be parsed correctly;
- critical paths such as `core` and `sys` are present.

This is useful for automated backup checks, but it does not replace a real restore test.

Analyze selected prefixes

To inspect only a specific part of the stored data, use `-filter` together with `-depth`:

```
stronghold operator raft snapshot inspect -depth 3 -filter=core raft.snap
```

This is useful when troubleshooting storage growth or identifying unusually large key groups.

Export JSON for scripts

For scripts and monitoring workflows, use JSON output:

```
stronghold operator raft snapshot inspect -format=json raft.snap
```

When combined with `-validate`, JSON output is convenient for automated processing with tools such as `jq`.

Important considerations

- `inspect` does not restore a snapshot or modify cluster state;
- checksum and structural validation do not guarantee that the snapshot is fully suitable for restore into a specific cluster;
- for full confidence, periodically run a test restore in a separate environment.

See also

- Save a storage snapshot
- Restore from a snapshot
- Automated snapshots

Restore from a snapshot

Snapshot restore is used when you need to bring Stronghold back to a previously saved state.

Before you start

- Bring the cluster back to an operational state after the incident: prepare fresh storage and obtain the temporary root token created during re-initialization.
- Make sure the target cluster uses integrated Raft storage as its primary storage backend.
- Copy the snapshot file to the node where you will run the restore.

- Make sure you have the original unseal or recovery keys. After restore, you need those original keys, not only the newly issued initialization keys.

> ⚠ Snapshot restore is available only for integrated Raft storage snapshots. If Stronghold runs on top of `etcd`, `postgresql`, or another external backend, use that storage system's own restore procedure.

Restore with CLI

Use the `-force` flag when restoring a snapshot:

```
d8 stronghold operator raft snapshot restore -force /tmp/snapshots/backup.snap
```

The `-force` flag is required because the current cluster state and the snapshot data belong to different storage states.

Restore with API

To restore through the API, use the `POST /sys/storage/raft/snapshot-force` endpoint:

```
curl \
  --request POST \
  --header "X-Vault-Token: ${VAULT_TOKEN}" \
  --data-binary @/tmp/snapshots/backup.snap \
  "${VAULT_ADDR}/v1/sys/storage/raft/snapshot-force"
```

After restore

After the snapshot is loaded, unseal Stronghold with the original keys:

```
d8 stronghold operator unseal
```

Practical recommendations

- Test restore procedures in a non-production environment instead of validating only snapshot creation.
- Record where backups are stored and who is responsible for key access.
- If you operate multiple cluster domains or replication setups, define the restore order for each of them separately.

Automated snapshots

Automated snapshots let Stronghold create scheduled backups of integrated Raft storage and write them either to local disk or to S3-compatible object storage.

⚠ Automated snapshots are available only when Stronghold uses integrated Raft storage. For `etcd`, `postgresql`, and other external backends, configure backup procedures provided by the storage system itself.

When to use them

Automated snapshots are designed for recurring backups without manual command execution. They are especially useful for production clusters where backup retention must be predictable and copies should be stored outside the cluster itself.

How they work

- You can create multiple named snapshot configurations.
- Each configuration defines the schedule, retention policy, and storage type.
- Supported storage types are `local` and `aws-s3`.
- For production environments, local storage is usually less suitable than external object storage: the active node can change over time, and backups are safer when stored outside the system they protect.

Create or update a configuration

Method	Path
POST	`/sys/storage/raft/snapshot-auto/config/:name`

The endpoint requires `sudo` privileges.

Core parameters

Parameter	Type	Required	Default	Description
`name`	String	Yes	—	Name of the configuration to create or update.
`interval`	Integer or string	Yes	—	Time between backups. You can specify seconds or Go duration format such as `24h`.
`retain`	Integer	No	`3`	Number of backups to keep. Older backups are deleted when the limit is exceeded.
`storage_type`	Immutable string	Yes	—	Storage type: `local` or `aws-s3`.
`path_prefix`	Immutable string	Yes	—	For `local`, the directory where snapshots are stored. For `aws-s3`, the object prefix inside the bucket.
`file_prefix`	Immutable string	No	`stronghold-snapshot`	Prefix for the file or object name.

Additional parameters for `local`

Parameter	Type	Required	Default	Description
`local_max_space`	Integer	No	`0`	Maximum number of bytes backup files with the specified `file_prefix` may use in the `path_prefix` directory. A value of `0` disables the check.

Additional parameters for `aws-s3`

Parameter	Type	Required	Default	Description
`aws_s3_bucket`	String	Yes	—	Bucket name used for backup storage.
`aws_s3_region`	String	No	—	Bucket region.
`aws_access_key_id`	String	No	—	Access key ID for the bucket.
`aws_secret_access_key`	String	No	—	Secret access key for the bucket.
`aws_s3_endpoint`	String	No	—	S3 service endpoint.
`aws_s3_disable_tls`	Boolean	No	—	Disables TLS for the S3 endpoint. Use only for testing.
`aws_s3_ca_certificate`	String	No	—	CA certificate for the S3 endpoint in PEM format.

Configuration examples

Local disk

The following `local-snapshot.json` file creates a configuration that saves a snapshot every 5 minutes into `/stronghold/data/backups`, keeps 4 copies, and uses the `main_stronghold` file prefix:

```
{
  "interval": "5m",
  "path_prefix": "/stronghold/data/backups",
  "file_prefix": "main_stronghold",
  "retain": "4",
  "storage_type": "local"
}
```

```
d8 stronghold write sys/storage/raft/snapshot-auto/config/my-local-snapshots @local-snapshot.json
```

> Before applying the configuration, make sure the directory from `path_prefix` exists and is writable. The `failed to create snapshot directory at destination` error usually means the directory is missing or unavailable.

S3-compatible storage

The following `minio-snapshot.json` file stores snapshots in S3-compatible object storage:

```
{
  "interval": "3m",
  "path_prefix": "snapshots",
  "file_prefix": "stronghold_backup",
  "retain": "15",
  "storage_type": "aws-s3",
  "aws_s3_bucket": "my_bucket",
  "aws_s3_endpoint": "minio.domain.ru",
  "aws_access_key_id": "<ACCESS_KEY>",
  "aws_secret_access_key": "<SECRET_ACCESS_KEY>"
}
```

```
d8 stronghold write sys/storage/raft/snapshot-auto/config/my-remote-snapshots @minio-snapshot.json
```

> Before applying the configuration, make sure the bucket already exists and the provided credentials have read and write permissions.

Update an existing configuration

To modify only selected fields, provide a partial JSON document:

```
{
  "interval": "3m",
  "retain": "10"
}
```

```
d8 stronghold write sys/storage/raft/snapshot-auto/config/my-local-snapshots @local-snapshot-update.json
```

List configurations

Method	Path
LIST	`/sys/storage/raft/snapshot-auto/config`

```
d8 stronghold list sys/storage/raft/snapshot-auto/config
```

Read configuration parameters

Method	Path
GET	`/sys/storage/raft/snapshot-auto/config/:name`

```
d8 stronghold read sys/storage/raft/snapshot-auto/config/my-remote-snapshots
```

For `aws-s3`, the response does not expose `aws_access_key_id` or `aws_secret_access_key`.

Delete a configuration

Method	Path
DELETE	`/sys/storage/raft/snapshot-auto/config/:name`

```
d8 stronghold delete sys/storage/raft/snapshot-auto/config/my-remote-snapshots
```

> Deleting an automated snapshot configuration does not remove existing snapshot files from local or object storage.

Read backup status

Method	Path
GET	`/sys/storage/raft/snapshot-auto/status/:name`

```
d8 stronghold read sys/storage/raft/snapshot-auto/status/my-remote-snapshots
```

Important status fields:

- `consecutive_errors`: number of backup errors in a row;
- `last_snapshot_end`: end time of the last successful snapshot;
- `last_snapshot_error`: text of the most recent error;
- `last_snapshot_start`: start time of the last completed backup;
- `last_snapshot_url`: location of the last successful snapshot;
- `next_snapshot_start`: next scheduled start time;
- `snapshot_start`: start time of the current backup job;
- `snapshot_url`: location of the currently written snapshot.

See also

- Save a storage snapshot
- Restore from a snapshot

KMS and HSM

HSM support

Stronghold supports Root key encryption using hardware security modules (HSM) such as TPM2, Rutoken ECP 3.0, JaCarta, and other devices that support the PKCS11 standard. For testing and development purposes, SoftHSM2 is also supported.

> ⚠ The HSM integration described on this page currently applies only to Standalone Stronghold installations. The examples below use a local configuration file and a `seal "pkcs11"` block in the standalone server configuration.

To use automatic unsealing via PKCS11, you must first create keys in the HSM and configure Stronghold to use them.

SoftHSM2

1. Install the required packages:

   ```
   apt install libsofthsm2 opensc
   ```

2. Create a configuration for SoftHSM2:

   ```
   mkdir /home/stronghold/softhsm
   cd softhsm
   echo "directories.tokendir = /home/stronghold/softhsm/" > /home/stronghold/
   softhsm2.conf
   ```

3. Generate keys in the HSM:

```
export SOFTHSM2_CONF=/home/stronghold/softhsm2.conf
HSMLIB="/usr/lib/x86_64-linux-gnu/softhsm/libsofthsm2.so"
pkcs11-tool --module $HSMLIB --init-token --so-pin 1234 --init-pin --pin 4321 --
label my_token --login

Using slot 0 with a present token (0x0)
Token successfully initialized
User PIN successfully initialized

pkcs11-tool --module $HSMLIB -L

Available slots:
Slot 0 (0xe6829d3): SoftHSM slot ID 0xe6829d3
  token label        : my_token
  token manufacturer : SoftHSM project
  token model        : SoftHSM v2
  token flags        : login required, rng, token initialized, PIN initialized,
other flags=0x20
  hardware version   : 2.6
  firmware version   : 2.6
  serial num         : 6a5468368e6829d3
  pin min/max        : 4/255
Slot 1 (0x1): SoftHSM slot ID 0x1
  token state:   uninitialized

pkcs11-tool --module $HSMLIB --login --pin 4321 --keypairgen --key-type rsa:4096
--label "vault-rsa-key"

Using slot 0 with a present token (0xe6829d3)
Key pair generated:
Private Key Object; RSA
  label:      vault-rsa-key
  Usage:      decrypt, sign, signRecover, unwrap
  Access:     sensitive, always sensitive, never extractable, local
Public Key Object; RSA 4096 bits
  label:      vault-rsa-key
  Usage:      encrypt, verify, verifyRecover, wrap
  Access:     local
```

4. Example Stronghold configuration (`config.hcl`):

```
api_addr="https://0.0.0.0:8200"
log_level = "warn"
ui = true

listener "tcp" {
  address = "0.0.0.0:8200"
  tls_cert_file = "/home/stronghold/cert.pem"
  tls_key_file  = "/home/stronghold/key.pem"
  #tls_require_and_verify_client_cert = true
  #tls_client_ca_file = "ca.crt"
  tls_disable = "false"
}
storage "raft" {
  path = "/home/stronghold/data"
}

seal "pkcs11" {
  lib = "/usr/lib/x86_64-linux-gnu/softhsm/libsofthsm2.so"
  token_label = "my_token"
  pin = "4321"
  key_label = "vault-rsa-key"
  rsa_oaep_hash = "sha1"
}
```

5. Start Stronghold:

```
export SOFTHSM2_CONF=/home/stronghold/softhsm2.conf
stronghold server -config config.hcl
```

Using Rutoken ECDSA 3.0

1. Download and install the `librtpkcs11ecp.so` library from https://www.rutoken.ru/.
2. Generate a key pair (public and private) in the token, which will be used for Root key encryption. This operation is performed with the `pkcs11-tool` utility from the `opensc` package:

```
HSMLIB="/usr/lib/librtpkcs11ecp.so"
pkcs11-tool --module $HSMLIB --init-token --so-pin 87654321 \
            --init-pin --pin 12345678 --label my_token --login
pkcs11-tool --module $HSMLIB --login --pin 12345678 --keypairgen \
            --key-type rsa:2048 --label "vault-rsa-key"

Using slot 0 with a present token (0x0)
Key pair generated:
Private Key Object; RSA
  label:      vault-rsa-key
  Usage:      decrypt, sign
  Access:     sensitive, always sensitive, never extractable, local
Public Key Object; RSA 2048 bits
  label:      vault-rsa-key
  Usage:      encrypt, verify
  Access:     local
```

3. Add the `pkcs11` unseal method to the Stronghold configuration:

```
...
seal "pkcs11" {
  lib = "/usr/lib/librtpkcs11ecp.so"
  token_label = "my_token"
  pin = "12345678"
  key_label = "vault-rsa-key"
}
```

4. Start Stronghold and run `init`:

```
systemctl start stronghold

stronghold operator init
```

5. Check the Stronghold status:

```
stronghold status

Key                      Value
---                      -----
Recovery Seal Type       shamir
Initialized              true
Sealed                   false
Total Recovery Shares    5
Threshold                3
Version                  1.15.2+hsm
Build Date               2025-04-03T13:06:02Z
Storage Type             raft
Cluster Name             stronghold-cluster-6586e287
Cluster ID               d7552773-2e8a-33b6-9c32-6749a4c9af13
HA Enabled               false
```

Migration from Shamir keys to HSM

1. Modify the Stronghold configuration by adding the `seal` block:

```
...
seal "pkcs11" {
  lib = "/usr/lib/librtpkcs11ecp.so"
  token_label = "my_token"
  pin = "12345678"
  key_label = "vault-rsa-key"
}
```

2. Restart Stronghold. The logs will show a message:

```
2025-04-03T17:08:13.431+0300 [WARN]  core: entering seal migration mode;
Stronghold will not automatically unseal even if using an autoseal:
from_barrier_type=shamir to_barrier_type=pkcs11
```

3. Perform the migration by entering the unseal keys:

```
stronghold operator unseal -migrate
```

After the migration is complete, Stronghold will automatically unseal using `pkcs11` on restart.

Migration from HSM to Shamir keys

1. Modify the configuration by adding the parameter `disabled = "true"` to the `seal` block:

```
...
seal "pkcs11" {
  lib = "/usr/lib/librtpkcs11ecp.so"
  token_label = "my_token"
  pin = "12345678"
  key_label = "vault-rsa-key"
  disabled = "true"
}
```

2. Restart Stronghold.
3. Perform the migration by entering the recovery keys:

```
stronghold operator unseal -migrate
```

After the migration is complete, Stronghold will require manual entry of unseal keys on each restart.

Yandex Cloud KMS

Stronghold supports `seal "yandexcloudkms"` for auto-unseal and root key protection through Yandex Cloud KMS.

> ⚠ `seal "yandexcloudkms"` is currently supported only for Standalone Stronghold installations.

Among cloud KMS seal integrations, Stronghold supports `yandexcloudkms`. `awskms` and `gcpckms` configurations are not supported in Stronghold.

What `seal "yandexcloudkms"` does

The `seal "yandexcloudkms"` configuration allows Stronghold to:

- use Yandex Cloud KMS for encryption and decryption operations related to the root key;
- automatically unseal after restart without manual unseal key entry;
- rely on an external KMS instead of locally managed key material.

If double encryption is also enabled, the external KMS must be available not only during unseal, but during normal Stronghold runtime as well.

Configuration examples

```
seal "yandexcloudkms" {
  kms_key_id  = "abj1abc23def456ghi78"
  oauth_token = "y0_AQAAAA..."
}
```

Example using a service account key:

```
seal "yandexcloudkms" {
  kms_key_id                = "abj1abc23def456ghi78"
  service_account_key_file = "/etc/stronghold/yc-sa-key.json"
}
```

`seal "yandexcloudkms"` parameters

Parameter	Required	Description
`kms_key_id`	yes	ID of the symmetric key in Yandex Cloud KMS.
`oauth_token`	no	OAuth token used for Yandex Cloud authentication.
`service_account_key_file`	no	Path to the JSON authorized key file of a service account.
`endpoint`	no	Custom Yandex Cloud API endpoint.
`disabled`	no	Used during migration from one seal mechanism to another.

Important details:

- `kms_key_id` is required;
- `oauth_token` and `service_account_key_file` are mutually exclusive;
- if neither `oauth_token` nor `service_account_key_file` is set, Stronghold attempts to use the VM service account through instance metadata;
- if `endpoint` is omitted, the default Yandex Cloud SDK endpoint is used.

Authentication order

For `yandexcloudkms`, authentication values are resolved in the following order:

1. Environment variables.
2. Stronghold configuration file values.
3. Yandex Cloud VM service account credentials.

This means environment variables take precedence over values from the `seal` block.

Environment variables

The following environment variables are supported:

- `YANDEXCLOUD_KMS_KEY_ID`
- `YANDEXCLOUD_OAUTH_TOKEN`
- `YANDEXCLOUD_SERVICE_ACCOUNT_KEY_FILE`

- `YANDEXCLOUD_ENDPOINT`

You can use them instead of the corresponding configuration file parameters, or combine them if that fits your operational model.

Access requirements

During initialization, Stronghold checks that the configured key exists and that the process has permission to perform encryption with it.

In practice, `seal "yandexcloudkms"` requires:

- an existing symmetric key in Yandex Cloud KMS;
- permissions to encrypt and decrypt with that key;
- valid authentication through an OAuth token, a service account key file, or a VM service account.

Practical recommendations

- For production environments, prefer a VM service account or a dedicated service account with minimum required permissions.
- Do not configure `oauth_token` and `service_account_key_file` at the same time.
- When rotating KMS keys, plan for rewrap operations and verify access to older key material versions.
- If you use double encryption, account for the availability of Yandex Cloud KMS during normal Stronghold runtime.

See also

- HSM support
- Double encryption
- Standalone configuration

Double encryption

`Seal wrap` is a mechanism that adds an extra encryption layer on top of the normal storage cryptographic barrier. In Stronghold documentation, it can be treated as **double encryption**: data is protected by Stronghold's built-in barrier and then additionally wrapped through the configured `seal` mechanism.

This approach is especially useful in environments with elevated key protection requirements and in infrastructures that rely on external KMS or HSM systems.

What seal wrap provides

The additional encryption layer can help with:

- stronger protection of critical security parameters;
- compliance-driven environments and stricter internal security policies;

- deployments where the most sensitive values must be additionally protected by an external `seal` mechanism;
- integrating Stronghold into infrastructures that use KMS or HSM systems.

How it works

With `seal wrap`, Stronghold stores selected data not only encrypted by the built-in storage barrier, but also wrapped again through a supporting `seal`.

In practice, this means:

- the root key and other critical parameters gain an extra protection layer;
- some values in auth methods and secrets engines can be stored in seal-wrapped form;
- the storage and replication model remains transparent to users, but reads and writes of such values can take longer.

What is usually seal-wrapped

Double encryption primarily applies to critical security parameters in the Stronghold core, including:

- the root key;
- the keyring;
- the recovery key and related unseal material;
- other internal critical Stronghold parameters.

In addition, supporting backends and plugins may use `seal wrap` for selected sensitive values. The original Vault documentation gives examples such as:

- auth methods including `LDAP`, `RADIUS`, `Okta`, and `AWS`;
- `PKI` for issuer data and keys;
- `SSH` for CA keys;
- `Transit` for keys and key policy data;
- `KMIP` for managed objects and CA keys.

Enabling and disabling

For supporting `seal` mechanisms, Vault enables `seal wrap` by default. In Stronghold, you can disable the extra wrapping for values other than the root key with:

```
disable_sealwrap = true
```

This parameter is described in the Standalone configuration.

Important details:

- disabling `seal wrap` does not disable root key protection by the `seal`;
- changing `disable_sealwrap` takes effect gradually as values are read or rewritten;
- enabling it again is also a lazy process rather than an immediate full rewrap of storage.

Sealwrap rewrap

When the `seal` configuration changes, Stronghold must wrap all seal-wrapped values again so they match the current encryption configuration. This process is called **seal rewrap**.

`Rewrap` is especially useful:

- after migrating to a new `seal`;
- after changing `seal` configuration parameters;
- after rotating key material in an external KMS or HSM.

Stronghold exposes the following API endpoints for rewrap management:

Method	Path
`GET`	`/sys/sealwrap/rewrap`
`POST`	`/sys/sealwrap/rewrap`

- `GET /sys/sealwrap/rewrap` returns the current process status and counters for processed entries;
- `POST /sys/sealwrap/rewrap` starts a rewrap process if one is not already running.

Example request:

```
curl \
  --header "X-Vault-Token: ${VAULT_TOKEN}" \
  --request POST \
  "${VAULT_ADDR}/v1/sys/sealwrap/rewrap"
```

During rewrap:

- the process runs in the background;
- progress is best monitored through `GET /sys/sealwrap/rewrap` and server logs;
- you should avoid making further `seal` configuration changes until it finishes.

Rewrap duration depends on the number of seal-wrapped values and the performance characteristics of the external `seal` backend.

Operational considerations

- `seal wrap` works only with `seal` mechanisms that support it;
- if you use an external HSM or remote KMS, seal-wrapped reads and writes can be noticeably slower than normal operations;
- this matters most for write-heavy workloads and environments with unstable connectivity to the HSM or KMS;
- if the external `seal` becomes unavailable during runtime, part of Stronghold functionality may be affected.

Seal wrap and replication

`Seal wrap` exists below replication logic. This means:

- replication carries the data and the requirement that a value should be seal-wrapped;
- the actual additional encryption is performed by the local `seal` on each cluster;
- different clusters can use different local HSM or KMS keys while keeping the same protection model.

Practical recommendations

- Use `seal wrap` when you need stronger protection for key material and other critical values.
- Evaluate the performance impact before enabling it broadly in production.
- For HSM integrations, verify that connectivity to the device is stable.
- Change `disable_sealwrap` in a controlled way, understanding that rewrapping happens gradually.

See also

- HSM support
- Standalone configuration

KV1/KV2 replication

KV1/KV2 replication

Documentation for KV1/KV2 replication in English is in development.

Namespaces

Namespaces

English documentation for namespaces is in development.

Cryptographic algorithms

Cryptographic algorithms

English documentation for cryptographic algorithms is in development.

Extensions and Integrations

Stronghold plugins

Stronghold supports three plugin types:

- `auth` for authentication methods;
- `secret` for secrets engines;
- `database` for database plugins.

Plugins can be:

- **built-in**, shipped with Stronghold and available without additional preparation;
- **external**, delivered separately and requiring operator actions before use.

External plugins run as separate processes and communicate with Stronghold over RPC. This isolates plugin execution from the main Stronghold process so a plugin failure should not crash the server itself.

Important considerations

- A plugin is uniquely identified by its type, name, and optionally version.
- The same plugin can be mounted at multiple `mount path`s.
- For external plugins, Stronghold validates the plugin binary with SHA256.
- A plugin must be registered in the plugin catalog before it can be enabled.

Standalone vs DKP

The loading flow for external plugins depends on how Stronghold is installed:

- in **Standalone**, the operator brings the plugin binary to the server, places it in `plugin_directory`, registers it, and then enables it;
- in **Deckhouse Kubernetes Platform**, the operator declares plugins in `ModuleConfig`, the platform downloads and validates them, places them into the Stronghold container, and the plugin is then registered and enabled in Stronghold.

Section pages

- Plugins in Standalone for plugin directory setup, registration, and mounting on a Linux server.
- Plugins in DKP for plugin delivery through `ModuleConfig` and later registration in Stronghold.

Plugins in Standalone

In a Standalone installation, external plugins are delivered to the server by the operator. After that, the plugin must be registered in the plugin catalog and only then enabled as a `secret`, `auth`, or `database` engine.

How external plugins work

External plugins are separate applications that Stronghold starts and communicates with over RPC.

This means:

- the plugin process does not share memory with the main Stronghold process;
- the plugin only receives explicitly provided interfaces and arguments;
- a plugin crash should not crash the whole Stronghold server.

Prerequisites

To use external plugins in a Standalone configuration, you must:

- set `api_addr` in the Stronghold configuration;
- configure `plugin_directory`;
- place plugin binaries in that directory;
- register the plugin in the plugin catalog.

Configure `plugin_directory`

Stronghold uses the `plugin_directory` setting in the configuration file to define the directory from which external plugins may be started.

Example:

```
api_addr = "https://stronghold.example.com:8200"

plugin_directory = "/opt/stronghold/plugins"
```

Important details:

- `plugin_directory` must not be a symbolic link;
- if `plugin_directory` is not set, external plugins cannot be used;
- the plugin binary must physically exist in that directory.

Register an external plugin

Before enabling an external plugin, register it in the plugin catalog so Stronghold can verify its authenticity and integrity.

Example:

```
PLUGIN_SHA=$(sha256sum /opt/stronghold/plugins/passthrough-plugin | awk '{print $1}')

stronghold plugin register \
  -sha256="${PLUGIN_SHA}" \
  secret \
  passthrough-plugin
```

If the plugin uses versioning, you can specify the version explicitly:

```
stronghold plugin register \
  -sha256="${PLUGIN_SHA}" \
  -version="v1.0.0" \
  secret \
  passthrough-plugin
```

Stronghold supports registering multiple versions of the same plugin.

Enable an external plugin

After registration, the plugin can be mounted at the desired `mount path`:

```
stronghold secrets enable -path=my-secrets passthrough-plugin
```

Example result:

```
Success! Enabled the passthrough-plugin secrets engine at: my-secrets/
```

After that, the plugin appears in the list of enabled engines.

Disable a plugin

Disabling an external plugin works the same way as for built-in engines:

```
stronghold secrets disable my-secrets
```

If you want to remove the external plugin from the catalog completely, deregister it after disabling:

```
stronghold plugin deregister secret passthrough-plugin
```

Plugin versions

A plugin can optionally report its own semantic version. If it does, Stronghold can use that information during registration.

If you specify a version manually:

- it should match semantic versioning;
- it is recommended to use the `v` prefix, for example `v1.0.0`;
- the same plugin type and name can coexist in multiple versions in the catalog.

If no version is specified when mounting, Stronghold chooses the plugin with the following precedence:

1. A plugin registered without a version.
2. The plugin with the most recent semantic version.
3. The built-in plugin.

File permissions and `mlock`

If Stronghold uses `mlock`, external plugin binaries may also need the capability to use `mlock`.

In such cases, you may need:

```
sudo setcap cap_ipc_lock=+ep /opt/stronghold/plugins/<plugin-binary>
```

Stronghold can also validate ownership and permissions of the plugin directory and plugin files when file permissions checks are enabled.

Practical recommendations

- Keep external plugins in a dedicated directory with restricted access.
- Verify SHA256 before registration.
- Do not use symbolic links for `plugin_directory`.
- For production, validate plugin compatibility with the installed Stronghold version in advance.

See also

- Plugins in DKP
- Standalone configuration

Plugins in DKP

In Deckhouse Kubernetes Platform, plugin delivery differs from a Standalone installation: the operator does not copy plugin binaries to the server manually, but declares the plugin list in `ModuleConfig`.

After that, the platform:

- downloads plugins from the specified URLs;
- verifies their checksums;
- places them into the Stronghold container;
- restarts Stronghold when the plugin list changes.

After delivery, the plugin still has to be registered in Stronghold and then enabled at the required path.

Configure the plugin list

The list of downloadable plugins is configured in `ModuleConfig`.

Example:

```
apiVersion: deckhouse.io/v1alpha1
kind: ModuleConfig
metadata:
  name: stronghold
spec:
  enabled: true
  version: 1
  settings:
    plugins:
      - name: "vault-plugin-secrets-github"
        url: "https://github.com/martinbaillie/vault-plugin-secrets-github/releases/
download/v2.3.2/vault-plugin-secrets-github-linux-amd64"
        sha256: "72cb1f2775ee2abf12ffb725e469d0377fe7bbb93cd7aaa6921c141eddecab87"
      - name: "vault-plugin-auth-any"
        url: "https://plugins.example.local/myplugins/vault-plugin-auth-any-v1.0.0-
linux-amd64"
        sha256: "c943b505b39b53e1f4cb07f2a3455b59eac523ebf600cb04813b9ad28a848b21"
        ignoreFailure: true
        insecureSkipVerify: false
        ca: |
          -----BEGIN CERTIFICATE-----
          MIIDDTCCAfWgAwIBAgIJAOb7PcmW8W9MMA0GCSqGSIb3DQEBCwUAMBQxEjAQBgNV
          BAMTCWxvY2FsaG9zdDAeFw0yNjA1MjAwMDAwMDBaFw0yNjA2MjAwMDAwMDBaMBQx
          EjAQBgNVBAMTCWxvY2FsaG9zdDCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoC
          ggEBAKHh4g5i1R+3+9XdG0RFLiX1x5T2PvQ92E/78vR6+Bn09+G0P+C6143+wLn
          j96/E8rHbHr4R6L0f62/OJZh8JnZ/qRqE1N8oNc06Vh9Y7X8EzF4nZ4KgX/3y6L
          vXD251Qm7g==
          -----END CERTIFICATE-----
```

Download parameters

- `name`: plugin binary name;
- `url`: URL used to download the plugin;
- `sha256`: SHA256 checksum of the plugin;
- `ignoreFailure`: allows Stronghold startup to continue even if this plugin could not be downloaded;
- `insecureSkipVerify`: disables TLS certificate verification for the remote server;
- `ca`: additional CA certificate used to validate TLS.

What happens when the plugin list changes

Adding or removing plugins triggers a Stronghold restart.

If a plugin cannot be downloaded or validated:

- Stronghold startup is blocked;
- plugins with `ignoreFailure: true` are the exception;
- if checksum validation fails, the plugin is treated as not loaded and is removed.

Air-gapped environments

In air-gapped environments where Stronghold does not have outbound internet access, you can host the plugin inside the DKP cluster itself.

One practical approach is:

1. Build or prepare an `nginx` container image that includes the plugin binary.
2. Run that container in Kubernetes.
3. Expose it through an internal Kubernetes `Service`.
4. Point `ModuleConfig` to a URL such as `http://<service>.<namespace>.svc.cluster.local/...` so the platform downloads the plugin through the internal Kubernetes service.

Example `ModuleConfig` fragment:

```
spec:
  settings:
    plugins:
      - name: "vault-plugin-auth-any"
        url: "http://plugin-repo.plugins.svc.cluster.local/vault-plugin-auth-any"
        sha256: "c943b505b39b53e1f4cb07f2a3455b59eac523ebf600cb04813b9ad28a848b21"
```

This approach lets you:

- avoid opening outbound internet access;
- keep plugin binaries inside the internal environment;
- manage plugin versions centrally through an internal service.

Register a plugin

After the plugin is delivered into the container, register it through the CLI:

```
PLUGIN_SHA=$(sha256sum <plugin_binary> | awk '{print $1;}')

d8 stronghold plugin register \
  -command <command_to_run_plugin_binary> \
  -sha256 "${PLUGIN_SHA}" \
  -version "<semantic_version>" \
  <plugin_type> \
  <plugin_name>
```

Example: register the secret plugin `mykv`:

```
d8 stronghold plugin register \
  -command mykvplugin \
  -sha256 "${PLUGIN_SHA}" \
  -version "v1.0.1" \
  secret \
  mykv
```

Enable a plugin

After registration, enable the plugin as a `secret` or `auth` engine:

```
d8 stronghold <secrets|auth> enable \
  -path <mount_path> \
  <plugin_name>
```

Meaning:

- `secrets`: for `secret`-type plugins;
- `auth`: for authentication plugins;
- `-path`: mount path;
- `plugin_name`: the registered plugin name.

Example:

```
d8 stronghold secrets enable -path test-kv mykv
```

Disable and remove a plugin

1. Disable all `secret` and `auth` methods that use the plugin.
2. Deregister the plugin:

```
d8 stronghold plugin deregister secret my-custom-plugin
```

1. Remove the plugin from `ModuleConfig`.

Practical recommendations

- For production, prefer an internal plugin repository or trusted artifact storage.
- Always define `sha256` and verify it against the actual binary.
- Use `ignoreFailure` only for non-critical plugins.
- Remember that changing the plugin list restarts Stronghold.

See also

- Plugins in Standalone

Usage

Deckhouse Stronghold user guide

This section is intended for Deckhouse Stronghold users.

The user guide currently includes the following sections:

- Project access - how to get access to a project.
- Core concepts - the main Stronghold concepts such as authentication, tokens, policies, identity, leases, and response wrapping.
- Authentication methods - ways to authenticate users and services in Stronghold.
- Managed Keys - how to use external cryptographic keys through `pkcs11` and `yandexcloudkms` with `ssh`, `pki`, and `transit`.
- Secrets engines - how to work with KV, PKI, Transit, LDAP, Kubernetes, databases, and other secrets engines.
- Stronghold Agent - automatic authentication, token handling, and secret delivery without changing application code.

Get started

Configuring access and first login

To work with Deckhouse Stronghold, get the access parameters from the administrator. Log in from the command line with the Deckhouse CLI `d8` utility or through the web interface if it is available in the installation.

Access parameters:

- Stronghold server address
- Authentication method and path
- Web interface address, if you log in through a browser
- Additional connection parameters, if they are required in your installation

Log in via the CLI

To configure access to your project from the command line, follow these steps.

1. Install the Deckhouse CLI `d8` utility.
2. Set the Stronghold server address in an environment variable:

```
export STRONGHOLD_ADDR=https://stronghold.domain.my
```

   Replace `https://stronghold.domain.my` with the address you received from the administrator.

3. Log in through OIDC. To do this, authenticate to Stronghold using the following command:

   ```
   d8 stronghold login -path=oidc_deckhouse -method=oidc -no-print
   ```

   This example uses OIDC authentication with the `oidc` method and the `oidc_deckhouse` path. If your installation uses different parameters, specify the values provided by the administrator.

   After you run the command, the Deckhouse CLI tool will open a browser to complete login through the configured OIDC provider. If the browser does not open automatically, copy the URL from the command output and open it manually.

4. Verify access using the following command:

   ```
   d8 stronghold status
   ```

To work with Deckhouse Stronghold, use commands in the following format:

```
d8 stronghold <command>
```

Log in through the web interface

If the Stronghold web interface is available in your installation, log in through a browser:

1. Open the Stronghold web interface address provided by the administrator.
2. Select a login method, for example, OIDC.
3. If required, specify the role name or any other parameters provided by the administrator.
4. Authenticate.

After a successful login, Stronghold opens the user interface with the corresponding access permissions.

After logging in, it's recommended that you create a test secret. The required steps are described in "Creating a test secret".

Creating a test secret

In this section, you can learn how to create a test secret in Deckhouse Stronghold, verify its contents, and change its value if needed.

This example uses:

- `kv` secrets engine
- `secret/my-first-secret` test path
- `username` and `password` test keys

If your installation uses a different mount path, replace `secret` with the path provided by the administrator.

> ⚠ Use only test values in this example. Do not store real passwords, tokens, or keys in test secrets.

To create a test secret, follow these steps:

1. Create a secret by writing test values to the selected path:

   ```
   d8 stronghold kv put secret/my-first-secret username=demo password=secret123
   ```

2. Verify the saved values:

   ```
   d8 stronghold kv get secret/my-first-secret
   ```

   Example output:

   ```
   ====== Data ======
   Key         Value
   ---         -----
   password    secret123
   username    demo
   ```

3. Edit the secret value by rewriting it to the same path:

   ```
   d8 stronghold kv put secret/my-first-secret username=demo password=new-secret
   ```

4. Make sure the value has been updated:

   ```
   d8 stronghold kv get secret/my-first-secret
   ```

> ℹ After verification, replace the test path and values with the parameters used in your project.

Auth methods

Authentication methods

Auth methods are the components in Stronghold that perform authentication and are responsible for assigning identity and a set of policies to a user. In all cases, Stronghold will enforce authentication as part of the request processing. In most cases, Stronghold will delegate the authentication administration and decision to the relevant configured external auth method (e.g., Kubernetes).

Stronghold also supports `WebAuthn` for passwordless authentication with `FIDO2` authenticators and `passkeys`.

Stronghold also supports `SAML` for browser-based Web SSO through an external `SAML 2.0` identity provider.

Having multiple auth methods enables you to use an auth method that makes the most sense for your use case of Stronghold and your organization.

For example, on developer machines, the Userpass is easiest to use. But for servers the AppRole method is the recommended choice.

To learn more about authentication, see the authentication concepts page.

Enabling/Disabling auth methods

Auth methods can be enabled/disabled using the CLI or the API.

```
d8 stronghold auth enable userpass
```

When enabled, auth methods are similar to secrets engines: they are mounted within the Stronghold mount table and can be accessed and configured using the standard read/write API. All auth methods are mounted underneath the `auth/` prefix.

By default, auth methods are mounted to `auth/<type>`. For example, if you enable “ldap”, then you can interact with it at `auth/ldap`. However, this path is customizable, allowing users with advanced use cases to mount a single auth method multiple times.

```
d8 stronghold auth enable -path=my-login userpass
```

When an auth method is disabled, all users authenticated via that method are automatically logged out.

External auth method considerations

When using an external auth method (e.g., Kubernetes), Stronghold will call the external service at the time of authentication and for subsequent token renewals. If the status of an entity changes in the external system (e.g., an account expires or is disabled), Stronghold denies requests to **renew** tokens associated with the entity. However, any existing token remain valid for the original grant period unless they are explicitly revoked within Stronghold. Operators should set appropriate token TTLs when using external authN methods.

OIDC providers

OIDC provider configuration

This page collects high-level setup steps on how to configure an OIDC application for various providers. For more general usage and operation information, see the Stronghold JWT/OIDC method documentation.

OIDC providers usually offer flexible configuration options, so it is recommended to review their recommended settings and best practices. The guides listed below will help you get started.

- Gitlab
- Keycloak
- Kubernetes

GitLab OIDC provider

Gitlab

1. Visit Settings > Applications.
2. Fill out Name and Redirect URIs.
3. Making sure to select the “openid” scope.
4. Copy client ID and secret.

Keycloak OIDC provider

Keycloak

1. Select/create a Realm and Client. Select a Client and visit Settings.
2. Client Protocol: openid-connect
3. Access Type: confidential
4. Standard Flow Enabled: On
5. Configure Valid Redirect URIs.
6. Save.
7. Visit Credentials. Select Client ID and Secret and note the generated secret.

Kubernetes as an OIDC provider

Kubernetes can act as an OIDC provider so that Stronghold can validate service account tokens using JWT auth or OIDC auth.

> The JWT auth mechanism does **not** use the Kubernetes API `TokenReview` for token validation. Instead, it uses public-key cryptography to verify the JWT contents. This means that tokens revoked by Kubernetes remain valid until they expire.
> To reduce this risk, use short TTLs for service account tokens or use Kubernetes auth, which uses the `TokenReview` API.

Using the discovery URL

When using discovery, you only need to specify the OIDC discovery URL. If the OIDC URL uses a custom certificate, you also need a trusted CA. This is the simplest configuration mode if the Kubernetes cluster meets the requirements.

The Kubernetes cluster must meet the following requirements:

- The `ServiceAccountIssuerDiscovery` feature gate must be enabled.
  - It is available starting from version 1.18 and enabled by default starting from version 1.20.
- The URL specified in the `--service-account-issuer` parameter of the `kube-apiserver` component must contain an address accessible from Stronghold. For most managed Kubernetes services, this address is public.
- Short-lived Kubernetes service account tokens must be used.
  - This behavior is enabled by default for tokens mounted into pods starting from Kubernetes 1.21.

To configure this mode, follow these steps:

1. Make sure the OIDC discovery URL does not require authentication, as described in the Kubernetes documentation.

   ```
   d8 k create clusterrolebinding oidc-reviewer  \
   --clusterrole=system:service-account-issuer-discovery \
   --group=system:unauthenticated
   ```

2. Determine the issuer URL for your cluster.

   ```
   ISSUER="$(d8 k get --raw /.well-known/openid-configuration | jq -r '.issuer')"
   ```

3. Enable and configure JWT auth in Stronghold.

   ```
   d8 stronghold auth enable jwt
   d8 stronghold write auth/jwt/config oidc_discovery_url="${ISSUER}"
   ```

4. Configure the required roles.

Using public keys to validate JWTs

This method can be useful if the Kubernetes API is not accessible from Stronghold or if you want a single JWT auth endpoint to serve multiple Kubernetes clusters using a public key chain.

The Kubernetes cluster must meet the following requirements:

- The `ServiceAccountIssuerDiscovery` feature gate must be enabled.
  - It is available starting from version 1.18 and enabled by default starting from version 1.20.
  - This requirement is optional if you have access to the `/etc/kubernetes/pki/sa.pub` file on the cluster master node. In this case, you can skip the steps for retrieving the key and converting it to PEM format because the key is already stored in the required format.
- Short-lived Kubernetes service account tokens must be used.
  - This behavior is enabled by default for tokens mounted into pods starting from Kubernetes 1.21.

To configure JWT auth using Kubernetes public keys, follow these steps:

1. Retrieve the public key used to sign service account tokens from your cluster's JWKS URI.

   ```
   # The jwks_uri is available in /.well-known/openid-configuration.
   d8 k get --raw "$(d8 k get --raw /.well-known/openid-configuration | jq -r '.jwk
   s_uri' | sed -r 's/.*\.[^/]+(.*)/\1/')"
   ```

2. Convert the keys from JWK format to PEM format. You can use a CLI utility or an online JWK-to-PEM converter.

3. Configure the JWT auth endpoint to use the retrieved keys.

   ```
   d8 stronghold write auth/jwt/config \
      jwt_validation_pubkeys="-----BEGIN PUBLIC KEY-----
   MIIBIjANBgkqhkiG9...
   -----END PUBLIC KEY-----","-----BEGIN PUBLIC KEY-----
   MIIBIjANBgkqhkiG9...
   -----END PUBLIC KEY-----"
   ```

4. Configure the required roles.

Creating roles and authenticating

After the JWT auth endpoint is configured, you can configure a role and authenticate. The examples below assume that you use the service account token available by default in all pods. If you need to control the API audience or TTL, see Specifying TTL and API audience.

Choose any value from the default API audiences. In these examples, the `aud` array contains only one API audience, `https://kubernetes.default.svc.cluster.local`.

To find the default API audience, create a new token. This requires the Deckhouse CLI `d8` tool v1.24.0 or later:

```
d8 k create token default | cut -f2 -d. | base64 --decode
{"aud":["https://kubernetes.default.svc.cluster.local"], ...
"sub":"system:serviceaccount:default:default"}
```

You can also read the token from a running pod:

```
d8 k exec my-pod -- cat /var/run/secrets/kubernetes.io/serviceaccount/token | cut -f2
-d. | base64 --decode
{"aud":["https://kubernetes.default.svc.cluster.local"], ...
"sub":"system:serviceaccount:default:default"}
```

Create a role for JWT auth that the `default` service account in the `default` namespace can use.

```
d8 stronghold write auth/jwt/role/my-role \
role_type="jwt" \
bound_audiences="<AUDIENCE-FROM-PREVIOUS-STEP>" \
user_claim="sub" \
bound_subject="system:serviceaccount:default:default" \
policies="default" \
ttl="1h"
```

Pods or clients that have access to the service account JWT can now authenticate with this token.

Authentication example using the Deckhouse CLI:

```
d8 stronghold write auth/jwt/login \
  role=my-role \
  jwt=@/var/run/secrets/kubernetes.io/serviceaccount/token
```

An equivalent HTTP query example:

```
curl \
  --fail \
  --request POST \
  --data '{"jwt":"<JWT-TOKEN-HERE>","role":"my-role"}' \
  "${STRONGHOLD_ADDR}/v1/auth/jwt/login"
```

Specifying TTL and API audience

If you need to specify a custom TTL or API audience for service account tokens, the following Pod manifest shows a volume mount that overrides the automatically mounted default token. This is

especially relevant if you cannot disable the `--service-account-extend-token-expiration` flag for `kube-apiserver` and want to use short TTLs.

When using the resulting token, set `bound_audiences=stronghold` when creating roles in JWT auth.

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  # The automountServiceAccountToken parameter is redundant in this example because
the
  # selected mountPath matches the default path. This override prevents creation of
the
  # default token. However, you can use this parameter to ensure that only one token
is
  # mounted if you choose a different mount path.
  automountServiceAccountToken: false
  containers:
    - name: nginx
      image: nginx
      volumeMounts:
        - name: custom-token
          mountPath: /var/run/secrets/kubernetes.io/serviceaccount
  volumes:
    - name: custom-token
      projected:
        defaultMode: 420
        sources:
          - serviceAccountToken:
              path: token
              expirationSeconds: 600 # Minimum TTL is 10 minutes.
              audience: stronghold   # Must match your role's `bound_audiences`
parameter.
          # The remaining parameters are added to mimic the standard token creation
behavior.
          # They create the same objects as when automountServiceAccountToken is
enabled.
          - configMap:
              name: kube-root-ca.crt
              items:
                - key: ca.crt
                  path: ca.crt
          - downwardAPI:
              items:
                - fieldRef:
                    apiVersion: v1
                    fieldPath: metadata.namespace
                  path: namespace
```

JWT method

JWT authentication

The authentication flow for roles of type “jwt” is simpler than OIDC since Stronghold only needs to validate the provided JWT.

JWT verification

JWT signatures will be verified against public keys from the issuer. This process can be done in three different ways, though only one method may be configured for a single backend:

- **Static Keys**. A set of public keys is stored directly in the backend configuration.
- **JWKS**. A JSON Web Key Set (JWKS) URL (and optional certificate chain) is configured. Keys will be fetched from this endpoint during authentication.
- **OIDC Discovery**. An OIDC Discovery URL (and optional certificate chain) is configured. Keys will be fetched from this URL during authentication. When OIDC Discovery is used, OIDC validation criteria (e.g. `iss`, `aud`, etc.) will be applied.

If multiple methods are needed, another instance of the backend can be mounted and configured at a different path.

Via the CLI

The default path is `/jwt`. If this auth method was enabled at a different path, specify `-path=/my-path` in the CLI.

```
d8 stronghold write auth/jwt/login role=demo jwt=...
```

Via the API

The default endpoint is `auth/jwt/login`. If this auth method was enabled at a different path, use that value instead of `jwt`.

```
$ curl \
    --request POST \
    --data '{"jwt": "your_jwt", "role": "demo"}' \
    http://127.0.0.1:8200/v1/auth/jwt/login
```

The response will contain a token at `auth.client_token`:

```
{
  "auth": {
    "client_token": "38fe9691-e623-7238-f618-c94d4e7bc674",
    "accessor": "78e87a38-84ed-2692-538f-ca8b9f400ab3",
    "policies": ["default"],
    "metadata": {
      "role": "demo"
    },
    "lease_duration": 2764800,
    "renewable": true
  }
}
```

Configuration

Auth methods must be configured in advance before users or machines can authenticate. These steps are usually completed by an operator or configuration management tool.

1. Enable the JWT auth method. Either the “jwt” or “oidc” name may be used. The backend will be mounted at the chosen name.

   ```
   $ d8 stronghold auth enable jwt
     or
   $ d8 stronghold auth enable oidc
   ```

2. Use the `/config` endpoint to configure Stronghold. To support JWT roles, either local keys, a JWKS URL, or an OIDC Discovery URL must be present. For OIDC roles, OIDC Discovery URL, OIDC Client ID and OIDC Client Secret are required.

   ```
   $ d8 stronghold write auth/jwt/config \
       oidc_discovery_url="https://myco.auth0.com/" \
       oidc_client_id="m5i8bj3iofytj" \
       oidc_client_secret="f4ubv72nfiu23hnsj" \
       default_role="demo"
   ```

   If you need to perform JWT verification with JWT token validation, then leave the `oidc_client_id` and `oidc_client_secret` blank.

   ```
   $ d8 stronghold write auth/jwt/config \
      oidc_discovery_url="https://MYDOMAIN.eu.auth0.com/" \
      oidc_client_id="" \
      oidc_client_secret="" \
   ```

3. Create a named role:

```
d8 stronghold write auth/jwt/role/demo \
    allowed_redirect_uris="http://localhost:8250/oidc/callback" \
    bound_subject="r3qX9DljwFIWhsiqwFiu38209F10atW6@clients" \
    bound_audiences="https://vault.plugin.auth.jwt.test" \
    user_claim="https://vault/user" \
    groups_claim="https://vault/groups" \
    policies=webapps \
    ttl=1h
```

This role authorizes JWTs with the given subject and audience claims, gives it the `webapps` policy, and uses the given user/groups claims to set up Identity aliases.

For the complete list of configuration options, please see the API documentation.

Bound claims

Once a JWT has been validated as being properly signed and not expired, the authorization flow will validate that any configured “bound” parameters match. In some cases there are dedicated parameters, for example `bound_subject`, which must match the JWT’s `sub` parameter. A role may also be configured to check arbitrary claims through the `bound_claims` map. The map contains a set of claims and their required values. For example, assume `bound_claims` is set to:

```
{
  "division": "Europe",
  "department": "Engineering"
}
```

Only JWTs containing both the “division” and “department” claims, and respective matching values of “Europe” and “Engineering”, would be authorized. If the expected value is a list, the claim must match one of the items in the list. To limit authorization to a set of email addresses:

```
{
  "email": ["fred@example.com", "julie@example.com"]
}
```

Bound claims can optionally be configured with globs.

Claims as metadata

Data from claims can be copied into the resulting auth token and alias metadata by configuring `claim_mappings`. This role parameter is a map of items to copy. The map elements are of the form: `"<JWT claim>":"<metadata key>"`. Assume `claim_mappings` is set to:

```
{
  "division": "organization",
  "department": "department"
}
```

This specifies that the value in the JWT claim “division” should be copied to the metadata key “organization”. The JWT “department” claim value will also be copied into metadata but will retain the key name. If a claim is configured in `claim_mappings`, it must existing in the JWT or else the authentication will fail.

Note: the metadata key name “role” is reserved and may not be used for claim mappings.

Claim specifications and JSON pointer

Some parameters (e.g. `bound_claims`, `groups_claim`, `claim_mappings`, `user_claim`) are used to point to data within the JWT. If the desired key is at the top of level of the JWT, the name can be provided directly. If it is nested at a lower level, a JSON Pointer may be used.

Assume the following JSON data to be referenced:

```
{
  "division": "North America",
  "groups": {
    "primary": "Engineering",
    "secondary": "Software"
  }
}
```

A parameter of `"division"` will reference “North America”, as this is a top level key. A parameter `"/groups/primary"` uses JSON Pointer syntax to reference “Engineering” at a lower level. Any valid JSON Pointer can be used as a selector. Refer to the JSON Pointer RFC for a full description of the syntax.

AppRole method

The `approle` auth method allows machines or *apps* to authenticate with Stronghold-defined *roles*. The open design of `AppRole` enables a varied set of workflows and configurations to handle large numbers of apps. This auth method is oriented to automated workflows (machines and services), and is less useful for human operators.

An “AppRole” represents a set of Stronghold policies and login constraints that must be met to receive a token with those policies. The scope can be as narrow or broad as desired. An AppRole can be created for a particular machine, or even a particular user on that machine, or a service spread across machines. The credentials required for successful login depend upon the constraints set on the AppRole associated with the credentials.

Authentication

Via the CLI:

The default path is `/approle`. If this auth method was enabled at a different path, specify `auth/my-path/login` instead.

```
$ d8 stronghold write auth/approle/login \
    role_id=db02de05-fa39-4855-059b-67221c5c2f63 \
    secret_id=6a174c20-f6de-a53c-74d2-6018fcceff64

Key                 Value
---                 -----
token               65b74ffd-842c-fd43-1386-f7d7006e520a
token_accessor      3c29bc22-5c72-11a6-f778-2bc8f48cea0e
token_duration      20m0s
token_renewable     true
token_policies      [default]
```

Via the API:

The default endpoint is `auth/approle/login`. If this auth method was enabled at a different path, use that value instead of `approle`.

```
$ curl \
    --request POST \
    --data '{"role_id":"988a9df-...","secret_id":"37b74931..."}' \
    http://127.0.0.1:8200/v1/auth/approle/login
```

The response will contain the token at `auth.client_token`:

```
{
  "auth": {
    "renewable": true,
    "lease_duration": 2764800,
    "metadata": {},
    "policies": ["default", "dev-policy", "test-policy"],
    "accessor": "5d7fb475-07cb-4060-c2de-1ca3fcbf0c56",
    "client_token": "98a4c7ab-b1fe-361b-ba0b-e307aacfd587"
  }
}
```

Configuration

Auth methods must be configured in advance before users or machines can authenticate. These steps are usually completed by an operator or configuration management tool.

Via the CLI:

1. Enable the AppRole auth method:

```
d8 stronghold auth enable approle
```

2. Create a named role:

```
$ d8 stronghold write auth/approle/role/my-role \
    secret_id_ttl=10m \
    token_num_uses=10 \
    token_ttl=20m \
    token_max_ttl=30m \
    secret_id_num_uses=40
```

> **Note:** If the token issued by your approle needs the ability to create child tokens, you will need to set token_num_uses to 0.

For the complete list of configuration options, please see the API documentation.

1. Fetch the RoleID of the AppRole:

```
$ d8 stronghold read auth/approle/role/my-role/role-id
role_id     db02de05-fa39-4855-059b-67221c5c2f63
```

2. Get a SecretID issued against the AppRole:

```
$ d8 stronghold write -f auth/approle/role/my-role/secret-id
secret_id               6a174c20-f6de-a53c-74d2-6018fcceff64
secret_id_accessor      c454f7e5-996e-7230-6074-6ef26b7bcf86
secret_id_ttl           10m
secret_id_num_uses      40
```

Via the API:

1. Enable the AppRole auth method:

```
$ curl \
    --header "X-Vault-Token: ..." \
    --request POST \
    --data '{"type": "approle"}' \
    http://127.0.0.1:8200/v1/sys/auth/approle
```

2. Create an AppRole with desired set of policies:

```
$ curl \
    --header "X-Vault-Token: ..." \
    --request POST \
    --data '{"policies": "dev-policy,test-policy"}' \
    http://127.0.0.1:8200/v1/auth/approle/role/my-role
```

3. Fetch the identifier of the role:

```
$ curl \
    --header "X-Vault-Token: ..." \
    http://127.0.0.1:8200/v1/auth/approle/role/my-role/role-id
```

The response will look like:

```
{
  "data": {
    "role_id": "988a9dfd-ea69-4a53-6cb6-9d6b86474bba"
  }
}
```

4. Create a new secret identifier under the role:

```
$ curl \
    --header "X-Vault-Token: ..." \
    --request POST \
     http://127.0.0.1:8200/v1/auth/approle/role/my-role/secret-id
```

The response will look like:

```
{
  "data": {
    "secret_id_accessor": "45946873-1d96-a9d4-678c-9229f74386a5",
    "secret_id": "37b74931-c4cd-d49a-9246-ccc62d682a25",
    "secret_id_ttl": 600,
    "secret_id_num_uses": 40
  }
}
```

Credentials/Constraints

RoleID

RoleID is an identifier that selects the AppRole against which the other credentials are evaluated. When authenticating against this auth method's login endpoint, the RoleID is a required argument (via `role_id`) at all times. By default, RoleIDs are unique UUIDs, which allow them to serve as secondary

secrets to the other credential information. However, they can be set to particular values to match introspected information by the client (for instance, the client's domain name).

SecretID

SecretID is a credential that is required by default for any login (via `secret_id`) and is intended to always be secret. (For advanced usage, requiring a SecretID can be disabled via an AppRole's `bind_secret_id` parameter, allowing machines with only knowledge of the RoleID, or matching other set constraints, to fetch a token). SecretIDs can be created against an AppRole either via generation of a 128-bit purely random UUID by the role itself (`Pull` mode) or via specific, custom values (`Push` mode). Similarly to tokens, SecretIDs have properties like usage-limit, TTLs and expirations.

Pull and push SecretID modes

If the SecretID used for login is fetched from an AppRole, this is operating in Pull mode. If a "custom" SecretID is set against an AppRole by the client, it is referred to as a Push mode. Push mode mimics the behavior of the deprecated App-ID auth method; however, in most cases Pull mode is the better approach. The reason is that Push mode requires some other system to have knowledge of the full set of client credentials (RoleID and SecretID) in order to create the entry, even if these are then distributed via different paths. However, in Pull mode, even though the RoleID must be known in order to distribute it to the client, the SecretID can be kept confidential from all parties except for the final authenticating client by using response wrapping.

Push mode is available for App-ID workflow compatibility, which in some specific cases is preferable, but in most cases Pull mode is more secure and should be preferred.

Further constraints

`role_id` is a required credential at the login endpoint. AppRole pointed to by the `role_id` will have constraints set on it. This dictates other `required` credentials for login. The `bind_secret_id` constraint requires `secret_id` to be presented at the login endpoint. Going forward, this auth method can support more constraint parameters to support varied set of Apps. Some constraints will not require a credential, but still enforce constraints for login. For example, `secret_id_bound_cidrs` will only allow logins coming from IP addresses belonging to configured CIDR blocks on the AppRole.

Username and password authentication method

The `userpass` auth method allows users to authenticate in Deckhouse Stronghold using a username and password.

Method features

When using the `userpass` method, keep the following features in mind:

- Usernames and passwords are specified directly in the authentication method at the path `auth/userpass/users/`.
- The `userpass` method cannot read usernames and passwords from an external source.

- Entered usernames are converted to lowercase. For example, `Mary` and `mary` are equivalent entries.

Configuration

To allow users to authenticate, configure the `userpass` method. These steps are usually completed by an operator or configuration management tool.

To configure authentication using the `userpass` method, follow these steps:

1. Enable the `userpass` auth method:

   ```
   d8 stronghold auth enable userpass
   ```

   The method will be enabled at the `auth/userpass` path.

   To enable the method at a different path, use the `-path` flag:

   ```
   d8 stronghold auth enable -path=<userpass_mount_path> userpass
   ```

2. If necessary, create a user who is allowed to authenticate:

   ```
   d8 stronghold write auth/<userpass_mount_path>/users/alice \
     password=Pass-123! \
     token_policies=admins
   ```

This creates user `alice` with password `Pass-123!` and the `admins` policy.

User authentication using the userpass method

Example command for user authentication using the `userpass` method:

```
d8 stronghold login -method=userpass username=alice password="Pass-123!"
```

User lockout

If a user provides incorrect credentials several times in a row, Stronghold stops validating them for a while and returns an access denied error. This behavior is called user lockout (`user_lockout`).

The time for which a user is locked out is called lockout duration (`lockout_duration`). After this time expires, the user can log in again.

The number of failed login attempts after which a user is locked out is called lockout threshold (`lockout_threshold`). The lockout threshold counter resets after a few minutes without login attempts or after a successful login. The interval after which the counter resets when there are no login attempts is called lockout counter reset (`lockout_counter_reset`).

User lockout helps reduce the risk of password guessing attacks.

The user lockout feature is enabled by default. Default values:

- `lockout_threshold`: 5 attempts.
- `lockout_duration`: 15 minutes.
- `lockout_counter_reset`: 15 minutes.

You can disable user lockout with the `auth tune` command by setting the `disable_lockout` parameter to `true`:

```
d8 stronghold auth tune -user-lockout-disable=true userpass
```

> ⚠ User lockout is supported only by the `userpass`, `ldap`, and `approle` auth methods.

Changing your own password

You can allow a user to change their own password in the `userpass` method. To do this, create a policy where the path to the user's password is formed based on the authenticated user's name.

Creating a policy

Use the following policy template:

```
path "auth/userpass/users/{{identity.entity.aliases.<accessor>.name}}/password" {
  capabilities = ["update"]
}
```

Get the `<accessor>` value with the command:

```
d8 stronghold read -field=accessor sys/auth/userpass
```

The `{{identity.entity.aliases.<accessor>.name}}` template automatically substitutes the authenticated user's name. Therefore, the path always points only to the current user's password.

The template works after logging in via the `userpass` method.

Allowing a user to change their own password

To allow a user to change their own password using the `userpass` method, follow these steps:

If the userpass method is already enabled

1. Get the unique method identifier:

   ```
   ACCESSOR=$(d8 stronghold read -field=accessor sys/auth/userpass)
   ```

2. Create a policy that allows a user authenticated via `userpass` to change their password:

   ```
   d8 stronghold policy write self-change-password - <<EOF
   path "auth/userpass/users/{{identity.entity.aliases.${ACCESSOR}.name}}/
   password" {
     capabilities = ["update"]
   }
   EOF
   ```

3. Assign the policy to the user you want to allow to change their password:

If the user exists

```
d8 stronghold write auth/userpass/users/alice/policies \
  token_policies="self-change-password"
```

This example will apply the `self-change-password` policy to the existing user `alice`.

If the user does not exist

```
d8 stronghold write auth/userpass/users/alice \
  password="OldPass-123!" \
  token_policies="self-change-password"
```

This example will create a user named `alice`, grant them authentication via `userpass`, and assign the `self-change-password` policy to them.

If the userpass method is not enabled

1. Enable the `userpass` method:

   ```
   d8 stronghold auth enable userpass
   ```

2. Get the unique method identifier:

   ```
   ACCESSOR=$(d8 stronghold read -field=accessor sys/auth/userpass)
   ```

3. Create a policy that allows a user authenticated via `userpass` to change their password:

   ```
   d8 stronghold policy write self-change-password - <<EOF
   path "auth/userpass/users/{{identity.entity.aliases.${ACCESSOR}.name}}/
   password" {
     capabilities = ["update"]
   }
   EOF
   ```

4. Assign the policy to the user you want to allow to change their password:

If the user exists

```
d8 stronghold write auth/userpass/users/alice/policies \
  token_policies="self-change-password"
```

This example will apply the `self-change-password` policy to the existing user `alice`

If the user does not exist

```
d8 stronghold write auth/userpass/users/alice \
  password="OldPass-123!" \
  token_policies="self-change-password"
```

This example will create a user named `alice`, grant them authentication via `userpass`, and assign the `self-change-password` policy to them.

Changing password as a user

After authentication, a user can change their password if they are allowed to do so.

Example command for a user to change their password:

```
d8 stronghold write auth/userpass/users/alice/password password="NewPass-456!"
```

If the user tries to change someone else's password, Stronghold returns a `permission denied` error. If a user enters incorrect login credentials when changing their password, their account may be locked.

Default password policy

If no custom password policy is assigned to the `userpass` method, Stronghold uses the default policy when creating a user or changing a user's password.

The default policy for the `userpass` method requires:

- `8` characters in the password
- At least one uppercase letter
- At least one lowercase letter
- At least one digit
- At least one hyphen (`-`)

To use a custom policy instead of the default password policy for the `userpass` method, run the following command (replace `policy_name` with the name of the desired policy):

```
d8 stronghold write auth/userpass/password-policy/{policy_name}
```

Token

Token auth method

The `token` auth method is built-in and automatically available at `/auth/token`. It allows users to authenticate using a token, as well to create new tokens, revoke secrets by token, and more.

When any other auth method returns an identity, Stronghold core invokes the token method to create a new unique token for that identity.

The token store can also be used to bypass any other auth method: you can create tokens directly, as well as perform a variety of other operations on tokens such as renewal and revocation.

Authentication

Via the CLI

```
d8 stronghold login token=<token>
```

Via the API

The token is set directly as a header for the HTTP API. The header should be either `X-Vault-Token: <token>` or `Authorization: Bearer <token>`.

LDAP

LDAP auth method

The `ldap` auth method allows authentication using an existing LDAP server and user/password credentials. This allows Stronghold to be integrated into environments using LDAP without duplicating the user/pass configuration in multiple places. Stronghold supports integration with various LDAP directory services, including the Russian directory service ALD Pro.

The mapping of groups and users in LDAP to Stronghold policies is managed by using the `users/` and `groups/` paths.

A note on escaping

It is up to the administrator to provide properly escaped DNs. This includes the user DN, bind DN for search, and so on.

The only DN escaping performed by this method is on usernames given at login time when they are inserted into the final bind DN, and uses escaping rules defined in RFC 4514.

Additionally, Active Directory has escaping rules that differ slightly from the RFC; in particular it requires escaping of '#' regardless of position in the DN (the RFC only requires it to be escaped when it is the first character), and '=', which the RFC indicates can be escaped with a backslash, but does not contain in its set of required escapes. If you are using Active Directory and these appear in your usernames, please ensure that they are escaped, in addition to being properly escaped in your configured DNs.

For reference, see RFC 4514 and this TechNet post on characters to escape in Active Directory.

Authentication

Via the CLI

```
$ d8 stronghold login -method=ldap username=mitchellh
Password (will be hidden):
Successfully authenticated! The policies that are associated
with this token are listed below:

admins
```

Via the API

```
$ curl \
    --request POST \
    --data '{"password": "foo"}' \
    http://127.0.0.1:8200/v1/auth/ldap/login/mitchellh
```

The response will be in JSON. For example:

```
{
  "lease_id": "",
  "renewable": false,
  "lease_duration": 0,
  "data": null,
  "auth": {
    "client_token": "c4f280f6-fdb2-18eb-89d3-589e2e834cdb",
    "policies": [
      "admins"
    ],
    "metadata": {
      "username": "mitchellh"
    },
    "lease_duration": 0,
    "renewable": false
  }
}
```

Configuration

Auth methods must be configured in advance before users or machines can authenticate. These steps are usually completed by an operator or configuration management tool.

1. Enable the ldap auth method:

   ```
   d8 stronghold auth enable ldap
   ```

2. Configure connection details for your LDAP server, information on how to authenticate users, and instructions on how to query for group membership. The configuration options are categorized and detailed below.

Connection parameters

- `url` (string, required) - The LDAP server to connect to. Examples: `ldap://ldap.myorg.com`, `ldaps://ldap.myorg.com:636`. This can also be a comma-delineated list of URLs, e.g. `ldap://ldap.myorg.com,ldaps://ldap.myorg.com:636`, in which case the servers will be tried in-order if there are errors during the connection process.

- `starttls` (bool, optional) - If true, issues a `StartTLS` command after establishing an unencrypted connection.
- `insecure_tls` - (bool, optional) - If true, skips LDAP server SSL certificate verification - insecure, use with caution!
- `certificate` - (string, optional) - CA certificate to use when verifying LDAP server certificate, must be x509 PEM encoded.
- `client_tls_cert` - (string, optional) - Client certificate to provide to the LDAP server, must be x509 PEM encoded.
- `client_tls_key` - (string, optional) - Client certificate key to provide to the LDAP server, must be x509 PEM encoded.

Binding parameters

There are two alternate methods of resolving the user object used to authenticate the end user: *Search* or *User Principal Name*. When using *Search*, the bind can be either anonymous or authenticated. User Principal Name is a method of specifying users supported by Active Directory. More information on UPN can be found in the Microsoft User Principal Name documentation.

Binding - authenticated search

- `binddn` (string, optional) - Distinguished name of object to bind when performing user and group search. Example: `cn=stronghold,ou=Users,dc=example,dc=com`
- `bindpass` (string, optional) - Password to use along with `binddn` when performing user search.
- `userdn` (string, optional) - Base DN under which to perform user search. Example: `ou=Users,dc=example,dc=com`
- `userattr` (string, optional) - Attribute on user attribute object matching the username passed when authenticating. Examples: `sAMAccountName`, `cn`, `uid`
- `userfilter` (string, optional) - Go template used to construct a ldap user search filter. The template can access the following context variables: [`UserAttr`, `Username`]. The default userfilter is `({{.UserAttr}}={{.Username}})` or `(userPrincipalName={{.Username}}@UPNDomain)` if the `upndomain` parameter is set. The user search filter can be used to restrict what user can attempt to log in. For example, to limit login to users that are not contractors, you could write `(&(objectClass=user)({{.UserAttr}}={{.Username}})(!(employeeType=Contractor)))`.

> ⚠ When specifying a `userfilter`, either the templated value `{{.UserAttr}}` or the literal value that matches `userattr` should be present in the filter to ensure that the search returns a unique result that takes `userattr` into consideration for entity alias mapping purposes and avoid possible collisions on login.

Binding - anonymous search

- `discoverdn` (bool, optional) - If true, use anonymous bind to discover the bind DN of a user
- `userdn` (string, optional) - Base DN under which to perform user search. Example: `ou=Users,dc=example,dc=com`

- `userattr` (string, optional) - Attribute on user attribute object matching the username passed when authenticating. Examples: `sAMAccountName`, `cn`, `uid`
- `userfilter` (string, optional) - Go template used to construct a ldap user search filter. The template can access the following context variables: [`UserAttr`, `Username`]. The default userfilter is `({{.UserAttr}}={{.Username}})` or `(userPrincipalName={{.Username}}@UPNDomain)` if the `upndomain` parameter is set. The user search filter can be used to restrict what user can attempt to log in. For example, to limit login to users that are not contractors, you could write `(&(objectClass=user)({{.UserAttr}}={{.Username}})(!(employeeType=Contractor)))`.
- `deny_null_bind` (bool, optional) - This option prevents users from bypassing authentication when providing an empty password. The default is `true`.
- `anonymous_group_search` (bool, optional) - Use anonymous binds when performing LDAP group searches. Defaults to `false`.

> ⚠ When specifying a `userfilter`, either the templated value `{{.UserAttr}}` or the literal value that matches `userattr` should be present in the filter to ensure that the search returns a unique result that takes `userattr` into consideration for entity alias mapping purposes and avoid possible collisions on login.

Alias dereferencing

- `dereference_aliases` (string, optional) - Control how aliases are dereferenced when performing the search. Possible values are: `never`, `finding`, `searching`, and `always`. `finding` will only dereference aliases during name resolution of the base. `searching` will dereference aliases after name resolution.

Binding - user principal name (AD)

- `upndomain` (string, optional) - userPrincipalDomain used to construct the UPN string for the authenticating user. The constructed UPN will appear as `[username]@UPNDomain`. Example: `example.com`, which will cause Stronghold to bind as `username@example.com`.

Group membership resolution

Once a user has been authenticated, the LDAP auth method must know how to resolve which groups the user is a member of. The configuration for this can vary depending on your LDAP server and your directory schema. There are two main strategies when resolving group membership - the first is searching for the authenticated user object and following an attribute to groups it is a member of. The second is to search for group objects of which the authenticated user is a member of. Both methods are supported.

- `groupfilter` (string, optional) - Go template used when constructing the group membership query. The template can access the following context variables: [`UserDN`, `Username`]. The default is `(|(memberUid={{.Username}})(member={{.UserDN}})(uniqueMember={{.UserDN}}))`, which is compatible with several common directory schemas. To support nested group resolution for

Active Directory, instead use the following query: `(&(objectClass=group)(member:1.2.840.113556.1.4.1941:={{.UserDN}}))`.

- `groupdn` (string, required) - LDAP search base to use for group membership search. This can be the root containing either groups or users. Example: `ou=Groups,dc=example,dc=com`
- `groupattr` (string, optional) - LDAP attribute to follow on objects returned by `groupfilter` in order to enumerate user group membership. Examples: for groupfilter queries returning *group* objects, use: `cn`. For queries returning *user* objects, use: `memberOf`. The default is `cn`.

Note: When using *Authenticated Search* for binding parameters (see above) the distinguished name defined for `binddn` is used for the group search. Otherwise, the authenticating user is used to perform the group search.

Other

- `username_as_alias` (bool, optional) - If set to true, forces the auth method to use the username passed by the user as the alias name.
- `max_page_size` (int, optional) - If set to a value greater than 0, the LDAP backend will use the LDAP server's paged search control to request pages of up to the given size. This can be used to avoid hitting the LDAP server's maximum result size limit. Otherwise, the LDAP backend will not use the paged search control.

Examples

Scenario 1

- LDAP server running on `ldap.example.com`, port 389.
- Server supports `STARTTLS` command to initiate encryption on the standard port.
- CA Certificate stored in file named `ldap_ca_cert.pem`
- Server is Active Directory supporting the userPrincipalName attribute. Users are identified as `username@example.com`.
- Groups are nested, we will use `LDAP_MATCHING_RULE_IN_CHAIN` to walk the ancestry graph.
- Group search will start under `ou=Groups,dc=example,dc=com`. For all group objects under that path, the `member` attribute will be checked for a match against the authenticated user.
- Group names are identified using their `cn` attribute.

```
$ d8 stronghold write auth/ldap/config \
    url="ldap://ldap.example.com" \
    userdn="ou=Users,dc=example,dc=com" \
    groupdn="ou=Groups,dc=example,dc=com" \
    groupfilter="(&(objectClass=group)(member:1.2.840.113556.1.4.1941:={{.UserDN}}))"
\
    groupattr="cn" \
    upndomain="example.com" \
    certificate=@ldap_ca_cert.pem \
    insecure_tls=false \
    starttls=true
...
```

Scenario 2

- LDAP server running on `ldap.example.com`, port 389.
- Server supports `STARTTLS` command to initiate encryption on the standard port.
- CA Certificate stored in file named `ldap_ca_cert.pem`
- Server does not allow anonymous binds for performing user search.
- Bind account used for searching is `cn=stronghold,ou=users,dc=example,dc=com` with password `My$ecrt3tP4ss`.
- User objects are under the `ou=Users,dc=example,dc=com` organizational unit.
- Username passed to stronghold when authenticating maps to the `sAMAccountName` attribute.
- Group membership will be resolved via the `memberOf` attribute of *user* objects. That search will begin under `ou=Users,dc=example,dc=com`.

```
$ d8 stronghold write auth/ldap/config \
    url="ldap://ldap.example.com" \
    userattr=sAMAccountName \
    userdn="ou=Users,dc=example,dc=com" \
    groupdn="ou=Users,dc=example,dc=com" \
    groupfilter="(&(objectClass=person)(uid={{.Username}}))" \
    groupattr="memberOf" \
    binddn="cn=stronghold,ou=users,dc=example,dc=com" \
    bindpass='My$ecrt3tP4ss' \
    certificate=@ldap_ca_cert.pem \
    insecure_tls=false \
    starttls=true
...
```

Scenario 3

- LDAP server running on `ldap.example.com`, port 636 (LDAPS)
- CA Certificate stored in file named `ldap_ca_cert.pem`
- User objects are under the `ou=Users,dc=example,dc=com` organizational unit.
- Username passed to Stronghold when authenticating maps to the `uid` attribute.

- User bind DN will be auto-discovered using anonymous binding.
- Group membership will be resolved via any one of `memberUid`, `member`, or `uniqueMember` attributes. That search will begin under `ou=Groups,dc=example,dc=com`.
- Group names are identified using the `cn` attribute.

```
$ d8 stronghold write auth/ldap/config \
    url="ldaps://ldap.example.com" \
    userattr="uid" \
    userdn="ou=Users,dc=example,dc=com" \
    discoverdn=true \
    groupdn="ou=Groups,dc=example,dc=com" \
    certificate=@ldap_ca_cert.pem \
    insecure_tls=false \
    starttls=true
...
```

LDAP group -> policy mapping

Next we want to create a mapping from an LDAP group to an Stronghold policy:

```
d8 stronghold write auth/ldap/groups/scientists policies=foo,bar
```

This maps the LDAP group “scientists” to the “foo” and “bar” Stronghold policies. We can also add specific LDAP users to additional (potentially non-LDAP) groups. Note that policies can also be specified on LDAP users as well.

```
d8 stronghold write auth/ldap/groups/engineers policies=foobar
d8 stronghold write auth/ldap/users/tesla groups=engineers policies=zoobar
```

This adds the LDAP user “tesla” to the “engineers” group, which maps to the “foobar” Stronghold policy. User “tesla” itself is associated with “zoobar” policy.

Finally, we can test this by authenticating:

```
$ d8 stronghold login -method=ldap username=tesla
Password (will be hidden):
Successfully authenticated! The policies that are associated
with this token are listed below:

default, foobar, zoobar
```

Note on policy mapping

It should be noted that user -> policy mapping happens at token creation time. And changes in group membership on the LDAP server will not affect tokens that have already been provisioned. To see these changes, old tokens should be revoked and the user should be asked to reauthenticate.

User lockout

If a user provides bad credentials several times in quick succession, Stronghold will stop trying to validate their credentials for a while, instead returning immediately with a permission denied error. We call this behavior “user lockout”. The time for which a user will be locked out is called “lockout duration”. The user will be able to login after the lockout duration has passed. The number of failed login attempts after which the user is locked out is called “lockout threshold”. The lockout threshold counter is reset to zero after a few minutes without login attempts, or upon a successful login attempt. The duration after which the counter will be reset to zero after no login attempts is called “lockout counter reset”. This can defeat both automated and targeted requests i.e, user-based password guessing attacks as well as automated attacks.

The user lockout feature is enabled by default. The default values for “lockout threshold” is 5 attempts, “lockout duration” is 15 minutes, “lockout counter reset” is 15 minutes.

The user lockout feature can be disabled using “auth tune” by setting `disable_lockout` to `true`.

> ⚠ This feature is only supported by the userpass, ldap, and approle auth methods.

SAML method

SAML auth method

The `saml` auth method lets users authenticate to Deckhouse Stronghold through an external `SAML 2.0` Identity Provider using the `Web SSO` profile.

This method is suitable for browser-based sign-in through the Stronghold UI and for custom integrations that start the SAML login flow through the HTTP API. Stronghold acts as a SAML Service Provider, validates the response from the Identity Provider, and then issues a Stronghold token according to the matched role.

How it works

The login flow has three stages:

1. A client requests `auth/<mount>/sso_service_url` and receives an IdP redirect URL plus a temporary `token_poll_id`.
2. The user completes authentication at the Identity Provider, which sends a signed SAML response to `auth/<mount>/callback`.

3. The client exchanges `token_poll_id` and `client_verifier` at `auth/<mount>/token` and receives a Stronghold token.

Stronghold supports two client modes:

- `browser` for UI-style login flows;
- `cli` for external tools that open the IdP page in a browser and then poll for the token.

Enable the method

```
d8 stronghold auth enable saml
```

By default the method is mounted at `auth/saml`. You can mount it at a custom path if needed:

```
d8 stronghold auth enable -path=corp-saml saml
```

Configuration

Before users can log in, configure Stronghold as a SAML Service Provider.

Main configuration parameters:

- `entity_id` is the Service Provider entity ID that must match the application settings in the Identity Provider.
- `acs_urls` is the list of allowed Assertion Consumer Service callback URLs.
- `default_role` is optional. If set, users can start login without explicitly passing a role.
- `idp_metadata_url` lets Stronghold fetch IdP settings from metadata.
- `idp_sso_url`, `idp_entity_id`, and `idp_cert` can be used instead of `idp_metadata_url` for manual configuration.
- `validate_response_signature` and `validate_assertion_signature` control signature validation. In production, enable both when the Identity Provider supports signing both objects.
- `verbose_logging` adds SAML exchange details to logs and should only be used for troubleshooting.

Configure via IdP metadata

```
d8 stronghold write auth/saml/config \
  entity_id="https://stronghold.example.com/v1/auth/saml" \
  acs_urls="https://stronghold.example.com/v1/auth/saml/callback" \
  idp_metadata_url="https://idp.example.com/app/stronghold/sso/saml/metadata" \
  default_role="employees" \
  validate_response_signature=true \
  validate_assertion_signature=true
```

Configure manually

Use manual configuration when metadata is unavailable:

```
d8 stronghold write auth/saml/config \
  entity_id="https://stronghold.example.com/v1/auth/saml" \
  acs_urls="https://stronghold.example.com/v1/auth/saml/callback" \
  idp_sso_url="https://idp.example.com/sso" \
  idp_entity_id="https://idp.example.com/entity" \
  idp_cert=@idp-signing-cert.pem \
  validate_response_signature=true \
  validate_assertion_signature=true
```

If multiple `acs_urls` are configured, the client must explicitly choose which one to use when starting the login flow.

Assertion consumer service URLs

The `acs_urls` parameter defines where the Identity Provider is allowed to send the SAML response after the user completes authentication.

When configuring `acs_urls`, make sure that each value:

- matches one of the callback URLs configured for the SAML application on the Identity Provider side;
- points to the Stronghold SAML callback endpoint `.../v1/auth/<mount>/callback` for the chosen mount path;
- uses `https://` in production environments.

If Stronghold is exposed through multiple public addresses, you can configure several ACS URLs:

```
d8 stronghold write auth/saml/config \
  entity_id="https://stronghold.example.com/v1/auth/saml" \
  acs_urls="https://primary.example.com/v1/auth/saml/callback,https://
secondary.example.com/v1/auth/saml/callback" \
  idp_metadata_url="https://idp.example.com/app/stronghold/sso/saml/metadata"
```

If you use namespaces, include the namespace path in the API URL or pass the `X-Vault-Namespace` header so that the resulting callback URL matches the real auth mount location.

Roles

SAML roles define which SAML subjects and attributes are allowed to authenticate and which token settings are applied after login.

Main role parameters:

- `bound_subjects` restricts allowed SAML subjects.

- `bound_subjects_type` defines matching mode: `string` or `glob`.
- `bound_attributes` requires specific assertion attributes and values.
- `bound_attributes_type` defines matching mode for attribute values: `string` or `glob`.
- `groups_attribute` maps a SAML attribute to Stronghold Identity group aliases.
- `alias_metadata` copies static metadata into the generated entity alias.
- token parameters such as `token_policies`, `token_ttl`, `token_max_ttl`, `token_period`, and `token_bound_cidrs` work the same way as in other auth methods.

Example role:

```
d8 stronghold write auth/saml/role/employees \
  bound_subjects="*@example.com" \
  bound_subjects_type="glob" \
  bound_attributes=department=platform \
  groups_attribute="memberOf" \
  token_policies="default,developers" \
  token_ttl="1h"
```

If your IdP sends multi-value attributes, `bound_attributes` can match any of the expected values. Matching is case-insensitive for attribute names.

Bound attributes

After the user is authenticated by the Identity Provider, Stronghold checks the role restrictions against the SAML assertion:

- `bound_subjects` matches the SAML subject;
- `bound_attributes` matches required assertion attributes and allowed values.

This lets you allow access only to selected users or groups from the Identity Provider. For example:

```
d8 stronghold write auth/saml/role/support \
  bound_subjects="*@example.com" \
  bound_subjects_type="glob" \
  bound_attributes=groups="support,engineering" \
  token_policies="support-ro"
```

The role above authorizes users whose subject ends with `@example.com` and whose `groups` attribute contains either `support` or `engineering`.

For Microsoft identity platforms, group membership is often sent in the attribute `http://schemas.microsoft.com/ws/2008/06/identity/claims/groups`. In that case, use that attribute name in `bound_attributes` and, if needed, in `groups_attribute`.

Link SAML groups to Stronghold Identity groups

If you want SAML group membership to grant Stronghold policies through Identity, use `groups_attribute` in the role and create matching external Identity groups and aliases.

Example flow:

1. Create an external Identity group with the required policies.
2. Read the auth mount accessor for the SAML auth method.
3. Create a group alias whose name matches the value coming from the SAML assertion.

Example commands:

```
d8 stronghold write identity/group \
  name="SamlDevelopers" \
  type="external" \
  policies="developers"
```

```
d8 stronghold auth list -format=json
```

```
d8 stronghold write identity/group-alias \
  name="engineering" \
  mount_accessor="<saml-mount-accessor>" \
  canonical_id="<identity-group-id>"
```

With this setup, a SAML login that returns `engineering` in the attribute referenced by `groups_attribute` will be linked to the external Identity group.

Sign in through the UI

The Stronghold UI supports SAML login directly.

Typical flow:

1. Select the `SAML` auth method in the login form.
2. Enter the role name if your mount does not use `default_role`.
3. Click `Sign In`.
4. Complete authentication at the external Identity Provider.
5. After the callback completes, Stronghold issues the token and signs the user in.

Start login through the API

The API is useful for custom portals, wrappers, and external CLI tools.

1. Generate a verifier and challenge

`client_challenge` must be a base64-encoded SHA-256 digest of `client_verifier`.

```
verifier="$(uuidgen)"
challenge="$(printf '%s' "$verifier" | openssl dgst -sha256 -binary | base64)"
```

2. Request the SSO URL

```
curl \
  --request POST \
  --data "{\"role\":\"employees\",\"client_challenge\":\"$challenge\",\"client_type\":
\"browser\"}" \
  https://stronghold.example.com/v1/auth/saml/sso_service_url
```

The response contains:

- `sso_service_url` for redirecting the user to the Identity Provider;
- `token_poll_id` for the final token exchange.

If several ACS URLs are configured, include `acs_url` in this request.

3. Exchange the temporary flow state for a token

After the IdP sends the user back and Stronghold accepts the SAML response, exchange the poll ID for a token:

```
curl \
  --request POST \
  --data "{\"token_poll_id\":\"<poll-id>\",\"client_verifier\":\"$verifier\"}" \
  https://stronghold.example.com/v1/auth/saml/token
```

The token is returned in `auth.client_token`.

Practical notes

- Use `https://` ACS URLs in production.
- If your IdP can sign both the SAML response and the assertion, enable both signature validation options.
- Do not leave `verbose_logging=true` enabled in production, because logs may contain sensitive SAML data.
- If a role has `token_bound_cidrs`, the final token exchange must come from an allowed client address.
- A SAML role must define at least one bound condition: `bound_subjects` or `bound_attributes`.
- If you configure multiple `acs_urls`, make sure the client sends the correct `acs_url` when starting the login flow.

Kubernetes auth method

The Kubernetes auth method can be used to authenticate with Stronghold using a Kubernetes Service Account Token. This method of authentication makes it easy to introduce a Stronghold token into a Kubernetes Pod.

You can also use a Kubernetes Service Account Token to log in via JWT auth. See the section "How to work with short-lived Kubernetes tokens" for a summary of why you might want to use JWT auth instead and how it compares to Kubernetes auth.

> If you are upgrading to Kubernetes v1.21+, ensure the configuration option `disable_iss_validation` is set to true. Assuming the default mount path, you can check with `d8 stronghold read -field disable_iss_validation auth/kubernetes/config`. See "Changes in JWT token behavior in Kubernetes 1.21+" for more details.

Authentication

Via the CLI

The auth method name depends on how it was created. When Stronghold is deployed as part of Deckhouse Kubernetes Platform (DKP), the `kubernetes_local` auth method is created automatically and configured for the Kubernetes cluster where Stronghold is running. If the Kubernetes auth method is created manually, the default name is `kubernetes` unless a different path is specified.

If the auth method was created under a different name, specify it using the `-path` parameter in the CLI. For example:

```
d8 stronghold write -path=your-path auth/kubernetes/login role=demo jwt=...
```

Via the API

Use the endpoint that matches the auth method name. When Stronghold is deployed as part of DKP, the automatically created auth method uses the `auth/kubernetes_local/login` endpoint. If the auth method was created under a different name, use the corresponding endpoint. The example below uses an auth method named `kubernetes`.

```
curl \
  --request POST \
  --data '{"jwt": "<your service account jwt>", "role": "demo"}' \
  https://stronghold.example.com/v1/auth/kubernetes/login
```

The response will contain a token at `auth.client_token`:

```
{
  "auth": {
    "client_token": "38fe9691-e623-7238-f618-c94d4e7bc674",
    "accessor": "78e87a38-84ed-2692-538f-ca8b9f400ab3",
    "policies": ["default"],
    "metadata": {
      "role": "demo",
      "service_account_name": "myapp",
      "service_account_namespace": "default",
      "service_account_secret_name": "myapp-token-pd21c",
      "service_account_uid": "aa9aa8ff-98d0-11e7-9bb7-0800276d99bf"
    },
    "lease_duration": 2764800,
    "renewable": true
  }
}
```


Configuration

Before users or applications can authenticate, the Kubernetes auth method must be configured. This is typically done by an operator or configuration management tool.

To configure authentication for another Kubernetes cluster, enable an additional instance of the Kubernetes auth method:

1. Enable the Kubernetes auth method:

   ```
   d8 stronghold auth enable kubernetes
   ```

2. Use the `/config` endpoint to configure Stronghold to talk to Kubernetes. Use `d8 k cluster-info` to validate the Kubernetes host address and TCP port.

   ```
   d8 stronghold write auth/kubernetes/config \
   token_reviewer_jwt="<your reviewer service account JWT>" \
   kubernetes_host=https://192.168.99.100:<your TCP port or blank for 443> \
   kubernetes_ca_cert=@ca.crt
   ```

   > ⚠ Stronghold uses the service account JWT token to authenticate with the Kubernetes API. Do not share this token with other applications or services, as they will be able to make requests to the Kubernetes API with the permissions of the corresponding service account.

3. Create a named role:

```
d8 stronghold write auth/kubernetes/role/demo \
  bound_service_account_names=myapp \
  bound_service_account_namespaces=default \
  policies=default \
  ttl=1h
```

This role authorizes the `myapp` service account in the `default` namespace and assigns it the default policy.

Changes in JWT token behavior in Kubernetes 1.21+

Starting in version Kubernetes 1.21, the Kubernetes feature gate `BoundServiceAccountTokenVolume` is enabled by default. As a result, the JWT tokens automatically mounted into containers have the following features:

- They have an expiry time.
- They are bound to the lifetime of a pod and service account.
- The value of the JWT's `iss` claim depends on the cluster's configuration.

The changes to token lifetime are important when configuring the `token_reviewer_jwt` option. If a short-lived token is used, Kubernetes will revoke it as soon as the pod or service account are deleted, or if the expiry time passes, and Stronghold will no longer be able to use the `TokenReview` API. For details on handling this change, see "How to work with short-lived Kubernetes tokens".

Kubernetes auth does not validate the `iss` field by default because Kubernetes already validates it when processing `TokenReview` requests.

How to work with short-lived Kubernetes tokens

There are a few different ways to configure auth for Kubernetes pods when default mounted pod tokens are short-lived, each with their own tradeoffs. This table summarizes the options, each of which is explained in more detail below.

Option	All tokens are short-lived	Can revoke tokens early	Other considerations
Use local token as reviewer JWT	Yes	Yes	Requires Stronghold to be deployed on the Kubernetes cluster
Use client JWT as reviewer JWT	Yes	Yes	Operational overhead
Use long-lived token as reviewer JWT	No	Yes	Simpler to configure
Use JWT auth instead	Yes	No	Client tokens cannot be revoked until they expire

> By default, Kubernetes currently extends the lifetime of admission injected service account tokens to a year to help smooth the transition to short-lived tokens. If you would like to disable this, set `--service-account-extend-token-expiration=false` for `kube-apiserver` or specify your own `serviceAccountToken` volume mount. See the "Specifying TTL and API audience" section for an example.

Using the Stronghold token as the reviewer JWT

When running Stronghold in a Kubernetes pod, Stronghold uses its local service account token as the reviewer JWT. Stronghold periodically re-reads the token file, which allows it to work with short-lived tokens.

To use the local token and CA certificate, omit `token_reviewer_jwt` and `kubernetes_ca_cert` when configuring the auth method. Stronghold will attempt to load them from `token` and `ca.crt` respectively inside the default mount folder `/var/run/secrets/kubernetes.io/serviceaccount/`.

```
d8 stronghold write auth/kubernetes/config \
kubernetes_host=https://$KUBERNETES_SERVICE_HOST:$KUBERNETES_SERVICE_PORT
```

Using the Stronghold client's JWT as the reviewer JWT

When configuring Kubernetes auth, you can omit the `token_reviewer_jwt`, and Stronghold will use the Stronghold client's JWT as its own auth token when communicating with the Kubernetes `TokenReview` API. If Stronghold is running in Kubernetes, you also need to set `disable_local_ca_jwt=true`.

This means Stronghold does not store any JWTs and allows you to use short-lived tokens everywhere but adds some operational overhead to maintain the cluster role bindings on the set of service accounts you want to be able to authenticate with Stronghold. Each client of Stronghold would need the `system:auth-delegator` ClusterRole:

```
d8 k create clusterrolebinding myapp-client-auth-delegator \
  --clusterrole=system:auth-delegator \
  --group=group1 \
  --serviceaccount=default:svcaccount1 \
  ...
```

Using long-lived tokens

You can create a long-lived token using the Kubernetes instructions and use that as the `token_reviewer_jwt`. In this example, the `myapp` service account would need the `system:auth-delegator` ClusterRole:

```
d8 k apply -f - <<EOF
apiVersion: v1
kind: Secret
metadata:
  name: myapp-k8s-auth-secret
  annotations:
    kubernetes.io/service-account.name: myapp
type: kubernetes.io/service-account-token
EOF
```

Using this maintains previous workflows but does not benefit from the improved security posture of short-lived tokens.

Using JWT auth

Kubernetes auth uses `TokenReview` API. However, the JWT tokens Kubernetes generates can also be verified using Kubernetes as an OIDC provider. The JWT auth method documentation has instructions for setting up JWT auth with Kubernetes as the OIDC provider.

This solution allows you to use short-lived tokens for all clients and removes the need for a `token_reviewer_jwt` configuration. However, the client tokens cannot be revoked before their TTL expires, so it is recommended to keep the TTL short with that limitation in mind.

Configuring Kubernetes

This auth method uses the `Kubernetes TokenReview API` to validate the provided JWT.

The service account used by this auth method must have permission to access the `TokenReview` API. The ClusterRoleBinding example below grants the required permissions:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: role-tokenreview-binding
  namespace: default
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: system:auth-delegator
subjects:
  - kind: ServiceAccount
    name: myapp-auth
    namespace: default
```

WebAuthn method

WebAuthn auth method

The `webauthn` auth method lets users authenticate to Deckhouse Stronghold with `FIDO2` authenticators and `passkeys`. `WebAuthn` support is available in Stronghold starting from version `1.17`.

This method is intended for passwordless sign-in to the Stronghold web UI and for custom web applications that use the Stronghold HTTP API. Make sure the user's browser and device support `WebAuthn` before enabling it for production use.

How it works

`WebAuthn` uses a two-step flow:

1. Stronghold returns registration or login options to the browser.
2. The browser talks to the user's authenticator.
3. The browser sends the result back to Stronghold, and Stronghold issues a token.

Login uses the `auth/<mount>/login/begin` and `auth/<mount>/login/finish` endpoints. Initial passkey enrollment uses `auth/<mount>/register/begin` and `auth/<mount>/register/finish`.

Configuration

Enable and configure the method before users can authenticate. The main parameters are:

- `rp_id` is the `Relying Party` identifier. This is typically the Stronghold DNS name. The host in `rp_id` must match the host of the origin used for authentication.
- `rp_display_name` is the human-readable service name shown by the browser and authenticator. If omitted, Stronghold uses `rp_id`.
- `rp_origins` is the list of allowed origins that can perform `WebAuthn` operations.
- `auto_registration` controls whether users can self-register passkeys. If `true` (default), users can enroll themselves. If `false`, users must be pre-created via the `user/` path.

Enable the method

```
d8 stronghold auth enable webauthn
```

By default the method is mounted at `auth/webauthn`. You can mount it at a custom path if needed:

```
d8 stronghold auth enable -path=my-passkeys webauthn
```

Configure the Relying Party

```
d8 stronghold write auth/webauthn/config \
  rp_id="stronghold.example.com" \
  rp_display_name="Deckhouse Stronghold" \
  rp_origins="https://stronghold.example.com"
```

Example with self-registration disabled:

```
d8 stronghold write auth/webauthn/config \
  rp_id="stronghold.example.com" \
  rp_display_name="Deckhouse Stronghold" \
  rp_origins="https://stronghold.example.com" \
  auto_registration=false
```

Pre-create a user

If `auto_registration=false`, an administrator must create the user in advance and define the token parameters:

```
d8 stronghold write auth/webauthn/user/alice \
  display_name="Alice Doe" \
  token_policies="developers" \
  token_ttl="1h"
```

The `auth/webauthn/user/<name>` path can be used to:

- pre-create a user for registration;
- update `display_name`;
- set token parameters such as `token_policies`, `token_ttl`, `token_max_ttl`, and `token_period`;
- inspect user metadata and the number of registered credentials;
- delete the user together with all enrolled passkeys.

Sign in through the UI

The Stronghold web UI supports `WebAuthn` directly.

To enroll a passkey for the first time:

1. Select the `WebAuthn` auth method.
2. Disable `Use passkey picker` so the `Username` field is shown.
3. Enter the username and click `Register`.
4. Confirm the operation on the device or in the passkey manager.

For later sign-ins, two modes are available:

- with `Use passkey picker` enabled, the browser shows discoverable credentials and the user does not need to enter a username;
- with it disabled, the user enters `Username` first and then the browser prompts for a matching passkey.

Registration and login through the API

The following flow is suitable for custom web applications.

Register a passkey

1. Request registration options:

```
curl \
  --request POST \
  --data '{"username":"alice"}' \
  https://stronghold.example.com/v1/auth/webauthn/register/begin
```

1. Pass the returned `publicKey` options to `navigator.credentials.create(...)` in the browser.
2. Finish registration by sending the credential back to Stronghold:

```
{
  "username": "alice",
  "credential": {
    "id": "...",
    "rawId": "...",
    "type": "public-key",
    "response": {
      "clientDataJSON": "...",
      "attestationObject": "..."
    }
  }
}
```

Send this payload to `POST /v1/auth/webauthn/register/finish`.

Login

1. Request login options:

```
curl \
  --request POST \
  --data '{"username":"alice"}' \
  https://stronghold.example.com/v1/auth/webauthn/login/begin
```

For passkey picker mode, `username` can be omitted.

1. Pass the returned options to `navigator.credentials.get(...)`.
2. Finish login by sending the authenticator response to `POST /v1/auth/webauthn/login/finish`:

```
{
  "username": "alice",
  "credential": {
    "id": "...",
    "rawId": "...",
    "type": "public-key",
    "response": {
      "clientDataJSON": "...",
      "authenticatorData": "...",
      "signature": "...",
      "userHandle": "..."
    }
  }
}
```

Stronghold returns the token in `auth.client_token`.

Practical notes

- Use `rp_origins` values that exactly match the real Stronghold UI or application URLs.
- In production, it is often better to set `auto_registration=false` and pre-create users with the required policies.
- `WebAuthn` is a good fit for passwordless UI sign-in, while machine-to-machine scenarios are usually better served by `AppRole`, `JWT`, or `Kubernetes`.

2FA / MFA

MULTIFACTOR Ldap Adapter

MULTIFACTOR LDAP Adapter is an LDAP proxy server developed and maintained by MULTIFACTOR. It is used to provide two-factor authentication for users of applications that use LDAP authentication. The system ensures multi-factor authentication and access control for any remote connections such as RDP, VPN, VDI, SSH, and others.

Configuring LDAP Adapter

How it works

Stronghold can perform two-factor authentication for users from an LDAP or Active Directory catalog:

1. The user connects to Stronghold and enters their username and password.
2. Stronghold connects via LDAP protocol to MULTIFACTOR LDAP Adapter.

3. The component verifies the user's login and password in Active Directory or another LDAP catalog and requests a second authentication factor.
4. The user confirms the access request using the selected authentication method.

Configuring MULTIFACTOR

1. Log in to the MULTIFACTOR management system. In the "Resources" section, create a new LDAP application. After creation, two parameters will become available (`NAS Identifier` and `Shared Secret`), which you will need in the following steps.
2. Download and install MULTIFACTOR LDAP Adapter.

Running LDAP Adapter in Kubernetes

To run the adapter, use the image `multifactor-ldap-adapter:3.0.7` and the following manifest:

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: ldap-adapter
spec:
  replicas: 1
  selector:
    matchLabels:
      app: ldap-adapter
  template:
    metadata:
      labels:
        app: ldap-adapter
    spec:
      containers:
      - image: registry.deckhouse.io/stronghold/multifactor/multifactor-ldap-adapter:
3.0.7
        name: ldap-adapter
        volumeMounts:
        - mountPath: /opt/multifactor/ldap/multifactor-ldap-adapter.dll.config
          name: config
          subPath: multifactor-ldap-adapter.dll.config
      volumes:
      - configMap:
          defaultMode: 420
          name: ldap-adapter
        name: config
---
apiVersion: v1
kind: Service
metadata:
  name: ldap-adapter
spec:
  ports:
  - port: 389
    protocol: TCP
    targetPort: 389
  selector:
    app: ldap-adapter
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: ldap-adapter
data:
  multifactor-ldap-adapter.dll.config: |
    <?xml version="1.0" encoding="utf-8"?>
```

```
    <configuration>
      <configSections>
        <section name="UserNameTransformRules"
type="MultiFactor.Ldap.Adapter.Configuration.UserNameTransformRulesSection,
multifactor-ldap-adapter" />
      </configSections>
      <appSettings>
        <add key="adapter-ldap-endpoint" value="0.0.0.0:389"/>
        <add key="ldap-server" value="ldap://ldap.example.com"/>
        <add key="ldap-service-accounts" value="CN=admin,DC=example,DC=com"/>
        <add key="ldap-base-dn" value="ou=Users,dc=example,dc=com"/>
        <add key="multifactor-api-url" value="https://api.multifactor.ru" />
        <add key="multifactor-nas-identifier" value="YOUR-NAS-IDENTIFIER" />
        <add key="multifactor-shared-secret" value="YOUR-NAS-SECRET" />
        <add key="logging-level" value="Debug"/>
      </appSettings>
    </configuration>
```

In the configuration, specify the address of your LDAP server and the values of `multifactor-nas-identifier` and `multifactor-shared-secret` obtained from the MULTIFACTOR management panel.

Available images:

- Ubuntu 24.04–based: `registry.deckhouse.io/stronghold/multifactor/multifactor-ldap-adapter:3.0.7`
- Alpine 3.22–based: `registry.deckhouse.io/stronghold/multifactor/multifactor-ldap-adapter:3.0.7-alpine`

Configuring Stronghold

To configure Stronghold, create and set up an `ldap` authentication method and specify the `ldap-adapter` address as the server. If you used the example manifest above to deploy the adapter, the address should be `ldap://ldap-adapter.default.svc`:

```
d8 stronghold auth enable ldap
d8 stronghold write auth/ldap/config url="ldap://ldap-adapter.default.svc" \
   binddn="cn=admin,dc=example,dc=com" bindpass="Password-1" \
   userdn="ou=Users,dc=example,dc=com" groupdn="ou=Groups,dc=example,dc=com" \
   username_as_alias=true
```

Testing with local OpenLDAP server

Below is an example manifest that you can use to deploy an OpenLDAP service in Kubernetes for testing purposes:

```
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: openldap
spec:
  replicas: 1
  selector:
    matchLabels:
      app: openldap
  template:
    metadata:
      labels:
        app: openldap
    spec:
      containers:
      - env:
        - name: LDAP_ADMIN_DN
          value: cn=admin,dc=example,dc=com
        - name: LDAP_ROOT
          value: dc=example,dc=com
        - name: LDAP_ADMIN_USERNAME
          value: admin
        - name: LDAP_ADMIN_PASSWORD
          value: Password-1
        image: bitnami/openldap:2.6.10
        name: openldap
---
apiVersion: v1
kind: Service
metadata:
  name: openldap
spec:
  ports:
  - name: p389
    port: 389
    protocol: TCP
    targetPort: 1389
  selector:
    app: openldap
```

After starting the container, create a user (for example, `alice` with the password `D3mo-Passw0rd`):

1. Access the OpenLDAP container:

   ```
   d8 k exec svc/openldap -it -- bash
   ```

2. Create the user with the following commands:

```
cd /tmp
cat << EOF > create_entries.ldif
dn: uid=alice,ou=users,dc=example,dc=com
objectClass: inetOrgPerson
objectClass: person
objectClass: top
cn: Alice
sn: User
userPassword: D3mo-Passw0rd
EOF

ldapadd -H ldap://openldap -cxD "cn=admin,dc=example,dc=com" \
        -w "Password-1" -f "create_entries.ldif"
```

You can now log in as the `alice` user using the password `D3mo-Passw0rd`. In the MULTIFACTOR management panel, a user named `alice` will appear under the “Users” section, where you can assign a second factor to them. In the future, confirmation will be required each time the user logs in to Stronghold. In addition to audit logs on the Stronghold side, the second-factor confirmation will also be recorded on the MULTIFACTOR side.

TOTP

Configuring TOTP

Stronghold supports verification of an additional authentication factor using Time-Based One-Time Passwords (TOTP), which are short-lived one-time codes. TOTP verification can be configured for a specific user or for the entire authentication method, including enforced mode.

To configure TOTP, follow these steps:

1. Enable the TOTP MFA method and obtain its ID:

```
TOTP_METHOD_ID=$(d8 stronghold write identity/mfa/method/totp \
    -format=json \
    generate=true \
    issuer=MyTOTP \
    period=30 \
    key_size=30 \
    algorithm=SHA256 \
    digits=6 | jq -r '.data.method_id')
echo $TOTP_METHOD_ID
```

2. If you need to enable (or recreate) TOTP MFA for a specific user, specify the user’s ID:

```
ENTITY_ID="f0075fa0-89ca-6235-5b90-b4420134cd36"
```

3. Generate a QR code for OTP configuration:

```
d8 stronghold write -field=barcode \
    /identity/mfa/method/totp/admin-generate \
    method_id=$TOTP_METHOD_ID entity_id=$ENTITY_ID \
    | base64 -d > /tmp/qr-code.png
```

If the user has access to the `identity/mfa/method/totp/generate` endpoint, they can obtain their own TOTP MFA configuration via the Stronghold UI using the method ID specified above.

Enabling MFA

As an example, let’s configure MFA verification for the Userpass authentication method.

1. Obtain the method’s accessor:

```
LDAP_ACCESSOR=$(d8 stronghold auth list -format=json \
    --detailed | jq -r '."userpass/".accessor')
echo $LDAP_ACCESSOR
```

2. Enable MFA:

```
d8 stronghold write /identity/mfa/login-enforcement/userpass-totp-enforcement \
    mfa_method_ids="$TOTP_METHOD_ID" \
    auth_method_accessors=$LDAP_ACCESSOR
```

3. Log in:

```
d8 stronghold login -method=userpass username=user password='My-Password-1234'
Initiating Interactive MFA Validation...
Enter the passphrase for methodID "22c35aa4-bf37-cf31-4187-c5a676c19aca" of type
"totp":
```

To disable MFA verification, run:

```
d8 stronghold delete identity/mfa/login-enforcement/userpass-totp-enforcement
```

Managed Keys

Managed Keys in Stronghold

`Managed Keys` let Stronghold use cryptographic keys stored in an external trusted system, such as an HSM or external KMS, without storing the private key material inside Stronghold itself.

This approach is useful when:

- private keys must remain outside Stronghold;
- signing or encryption operations must be performed by an external system;
- security or compliance requirements prohibit exporting key material.

In Stronghold, a managed key is a named record managed through the `sys/managed-keys/<type>/<name>` API.

How it works

Stronghold stores the configuration needed to access an external key, but not the private key itself. When a secrets engine needs to perform signing, signature verification, encryption, or decryption, it calls the corresponding managed key, which then delegates the operation to the external backend.

This means:

- key material remains in the external system;
- Stronghold uses the managed key as an abstraction over the external backend;
- multiple managed keys can be configured for the same backend if you need different keys or different access policies.

Supported backends

Stronghold supports the following managed key types:

- `pkcs11`
- `yandexcloudkms`

`pkcs11`

The `pkcs11` type is used for HSMs and PKCS#11-compatible libraries. A managed key of this type references a previously defined `kms_library "pkcs11"` stanza in Stronghold server configuration.

A typical workflow is:

- configure `kms_library "pkcs11"` in the server configuration;
- register a managed key through `sys/managed-keys/pkcs11/<name>`;
- optionally verify key access using `test/sign`;
- allow the key for the required mount path.

Example:

```
stronghold write sys/managed-keys/pkcs11/my-hsm-key \
  library=softhsm \
  token_label=managed-keys \
  pin=1234 \
  key_label=my-signing-key \
  usages=sign,verify
```

`yandexcloudkms`

The `yandexcloudkms` type is used for integration with Yandex Cloud KMS.

Its configuration uses:

- `kms_key_id`
- `oauth_token` or `service_account_key_json`
- `endpoint` when needed

For authentication to Yandex Cloud KMS, Stronghold can use one of the following options:

- `oauth_token`
- `service_account_key_json`
- the service account attached to the virtual machine in Yandex Cloud

If neither `oauth_token` nor `service_account_key_json` is provided, Stronghold attempts to use the virtual machine service account through the standard instance credentials flow. This is similar to common Vault patterns where cloud KMS integrations can rely on the identity of the running VM instance.

Example:

```
stronghold write sys/managed-keys/yandexcloudkms/my-yc-key \
  kms_key_id=<kms_key_id> \
  oauth_token=<oauth_token> \
  usages=sign,verify
```

Example using the virtual machine service account:

```
stronghold write sys/managed-keys/yandexcloudkms/my-yc-key \
  kms_key_id=<kms_key_id> \
  usages=sign,verify
```

> The `oauth_token` and `service_account_key_json` parameters are mutually exclusive. If one of them is specified, it is used explicitly for authentication to Yandex Cloud.

> For `yandexcloudkms`, the set of allowed operations depends on the key itself and its configuration. If the managed key is going to be used by `pki` or `ssh`, it must support signing. For `transit`, the required operations depend on the selected usage scenario.

Which secrets engines can use Managed Keys

In Stronghold, managed keys can be used by:

- `SSH`
- `PKI`
- `Transit`

SSH

The SSH secrets engine can use a managed key as its certificate authority key and use it to sign SSH certificates.

This is useful when the SSH CA key must remain in an external HSM or KMS.

PKI

The PKI secrets engine can use a managed key for root and intermediate CA generation and for certificate signing.

This is one of the primary managed key use cases in Stronghold. For PKI, it is also important to allow the managed key for the specific PKI mount if the key was not created with `any_mount=true`.

Example:

```
stronghold secrets tune -allowed-managed-keys=my-hsm-key pki/
```

Transit

The Transit secrets engine can use a managed key as an external cryptographic key.

Depending on backend capabilities and the key type, this can allow:

- signing;
- signature verification;
- encryption;
- decryption.

For Transit, the managed key is specified when creating or rotating a transit key.

Namespaces and scope

A managed key is bound to a specific namespace. The secrets engine that uses the key must exist in the same namespace as the managed key itself.

If the key is not created with `any_mount=true`, you must explicitly allow its use for a given mount path.

Basic operations

List managed keys of a given type:

```
stronghold list sys/managed-keys/pkcs11
```

Read key configuration:

```
stronghold read sys/managed-keys/pkcs11/my-hsm-key
```

Verify key accessibility with a test signature:

```
stronghold write sys/managed-keys/pkcs11/my-hsm-key/test/sign
```

Delete a managed key:

```
stronghold delete sys/managed-keys/pkcs11/my-hsm-key
```

Practical recommendations

- Use `pkcs11` when keys live in a local or network-attached HSM exposed through a PKCS#11 library.
- Use `yandexcloudkms` when keys are managed in Yandex Cloud KMS.
- Restrict `usages` to the minimum required operations.
- If the key should not be available to every mount, do not enable `any_mount` and configure `allowed-managed-keys` explicitly.
- Before attaching a key to `pki`, `ssh`, or `transit`, verify it with `test/sign` or another appropriate validation flow.

Related sections

- PKI secrets engine
- Transit secrets engine
- Signed SSH certificates

Secrets engines

Secrets engines

Secrets engines are components which store, generate, or encrypt data. Secrets engines are incredibly flexible, so it is easiest to think about them in terms of their function. Secrets engines are provided some set of data, they take some action on that data, and they return a result.

Some secrets engines simply store and read data - like the K/V plugin. Other secrets engines connect to other services and generate dynamic credentials on demand. Other secrets engines provide encryption as a service, totp generation, certificates, and much more.

Secrets engines are enabled at a **path** in Stronghold. When a request comes to Stronghold, the router automatically routes anything with the route prefix to the secrets engine. In this way, each secrets engine defines its own paths and properties. To the user, secrets engines behave similar to a virtual filesystem, supporting operations like read, write, and delete.

Secrets engines lifecycle

Most secrets engines can be enabled, disabled, tuned, and moved via the CLI or API.

- `Enable` - This enables a secrets engine at a given path. With a few exceptions, secrets engines can be enabled at multiple paths. Each secrets engine is isolated to its path. By default, they are enabled at their “type” (e.g. “kv” enables at `kv/`).

> **Case-sensitive:** The path where you enable secrets engines is case-sensitive. For example, the KV secrets engine enabled at `kv/` and `KV/` are treated as two distinct instances of KV secrets engine.

- `Disable` - This disables an existing secrets engine. When a secrets engine is disabled, all of its secrets are revoked (if they support it), and all the data stored for that engine in the physical storage layer is deleted.
- `Move` - This moves the path for an existing secrets engine. This process revokes all secrets, since secret leases are tied to the path where they were created. The configuration data stored for the engine persists through the move.
- `Tune` - This tunes global configuration for the secrets engine such as the TTLs.

Once a secrets engine is enabled, you can interact with it directly at its path according to its own API. Use `d8 stronghold path-help` to determine the paths it responds to.

Note that mount points cannot conflict with each other in Stronghold. There are two broad implications of this fact. The first is that you cannot have a mount which is prefixed with an existing mount. The second is that you cannot create a mount point that is named as a prefix of an existing mount. As an example, the mounts `foo/bar` and `foo/baz` can peacefully coexist with each other whereas `foo` and `foo/baz` cannot

Barrier view

Secrets engines receive a *barrier view* to the configured Stronghold physical storage. This is a lot like a chroot.

When a secrets engine is enabled, a random UUID is generated. This becomes the data root for that engine. Whenever that engine writes to the physical storage layer, it is prefixed with that UUID folder.

Since the Stronghold storage layer doesn't support relative access (such as `../`), this makes it impossible for an enabled secrets engine to access other data.

This is an important security feature in Stronghold - even a malicious engine cannot access the data from any other engine.

KV Secrets engine

KV secrets engine

The `kv` secrets engine is a generic Key-Value store used to store arbitrary secrets within the configured physical storage for Stronghold. This backend can be run in one of two modes; either it can be configured to store a single value for a key or, versioning can be enabled and a configurable number of versions for each key will be stored.

KV version 1

When running the `kv` secrets backend non-versioned, only the most recently written value for a key will be preserved. The benefits of non-versioned `kv` is a reduced storage size for each key since no additional metadata or history is stored. Additionally, requests going to a backend configured this way will be more performant because for any given request there will be fewer storage calls and no locking.

KV version 2

When running v2 of the `kv` backend a key can retain a configurable number of versions. This defaults to 10 versions. The older versions' metadata and data can be retrieved. Additionally, Check-and-Set operations can be used to avoid overwriting data unintentionally.

When a version is deleted the underlying data is not removed, rather it is marked as deleted. Deleted versions can be undeleted. To permanently remove a version's data the destroy command or API endpoint can be used. Additionally all versions and metadata for a key can be deleted by deleting on the metadata command or API endpoint. Each of these operations can be ACL'ed differently, restricting who has permissions to soft delete, undelete, or fully remove data.

KV/v1

The `kv` secrets engine is used to store arbitrary secrets within the configured physical storage for Stronghold.

Writing to a key in the `kv` backend will replace the old value; sub-fields are not merged together.

Key names must always be strings. If you write non-string values directly via the CLI, they will be converted into strings. However, you can preserve non-string values by writing the key/value pairs to Stronghold from a JSON file or using the HTTP API.

This secrets engine honors the distinction between the `create` and `update` capabilities inside ACL policies.

> ⚠ **Note**: Path and key names are *not* obfuscated or encrypted; only the values set on keys are. You should not store sensitive information as part of a secret's path.

Setup

To enable a version 1 kv store:

```
d8 stronghold secrets enable -version=1 kv
```

Usage

After the secrets engine is configured and a user/machine has an Stronghold token with the proper permission, it can generate credentials. The `kv` secrets engine allows for writing keys with arbitrary values.

1. Write arbitrary data:

   ```
   $ d8 stronghold kv put kv/my-secret my-value=s3cr3t
   Success! Data written to: kv/my-secret
   ```

2. Read arbitrary data:

   ```
   $ d8 stronghold kv get kv/my-secret
   Key                 Value
   ---                 -----
   my-value            s3cr3t
   ```

3. List the keys:

   ```
   $ d8 stronghold kv list kv/
   Keys
   ----
   my-secret
   ```

4. Delete a key:

   ```
   $ d8 stronghold kv delete kv/my-secret
   Success! Data deleted (if it existed) at: kv/my-secret
   ```

You can also use Stronghold's password policy feature to generate arbitrary values.

1. Write a password policy:

```
$ d8 stronghold write sys/policies/password/example policy=-<<EOF

  length=20

  rule "charset" {
    charset = "abcdefghij0123456789"
    min-chars = 1
  }

  rule "charset" {
    charset = "!@#$%^&*STUVWXYZ"
    min-chars = 1
  }

EOF
```

2. Write data using the `example` policy:

```
$ d8 stronghold kv put kv/my-generated-secret \
    password=$(d8 stronghold read -field password sys/policies/password/example/
generate)
```

3. Read the generated data:

```
$ d8 stronghold kv get kv/my-generated-secret
====== Data ======
Key         Value
---         -----
password    ^dajd609Xf8Zhac$dW24
```

TTLs

Unlike other secrets engines, the KV secrets engine does not enforce TTLs for expiration. Instead, the `lease_duration` is a hint for how often consumers should check back for a new value.

If provided a key of `ttl`, the KV secrets engine will utilize this value as the lease duration:

```
$ d8 stronghold kv put kv/my-secret ttl=30m my-value=s3cr3t
Success! Data written to: kv/my-secret
```

Even with a `ttl` set, the secrets engine *never* removes data on its own. The `ttl` key is merely advisory.

When reading a value with a `ttl`, both the `ttl` key *and* the refresh interval will reflect the value:

```
$ d8 stronghold kv get kv/my-secret
Key                 Value
---                 -----
my-value            s3cr3t
ttl                 30m
```

KV/v2

The `kv` secrets engine is used to store arbitrary secrets within the configured physical storage for Stronghold.

Key names must always be strings. If you write non-string values directly via the CLI, they will be converted into strings. However, you can preserve non-string values by writing the key/value pairs to Stronghold from a JSON file or using the HTTP API.

This secrets engine honors the distinction between the `create` and `update` capabilities inside ACL policies. The `patch` capability is also supported which is used to represent partial updates whereas the `update` capability represents full overwrites.

Setup

Most secrets engines must be configured in advance before they can perform their functions. These steps are usually completed by an operator or configuration management tool.

A v2 `kv` secrets engine can be enabled by:

```
d8 stronghold secrets enable -version=2 kv
```

Or, you can pass `kv-v2` as the secrets engine type:

```
d8 stronghold secrets enable kv-v2
```

Additionally, when running a dev-mode server, the v2 `kv` secrets engine is enabled by default at the path `secret/` (for non-dev servers, it is currently v1). It can be disabled, moved, or enabled multiple times at different paths. Each instance of the KV secrets engine is isolated and unique.

Upgrading from version 1

An existing version 1 kv store can be upgraded to a version 2 kv store via the CLI or API, as shown below. This will start an upgrade process to upgrade the existing key/value data to a versioned format. The mount will be inaccessible during this process. This process could take a long time, so plan accordingly.

Once upgraded to version 2, the former paths at which the data was accessible will no longer suffice. You will need to adjust user policies to add access to the version 2 paths as detailed in the ACL Rules

section below. Similarly, users/applications will need to update the paths at which they interact with the kv data once it has been upgraded to version 2.

An existing version 1 kv can be upgraded to a version 2 KV store with the CLI command:

```
$ d8 stronghold kv enable-versioning secret/
Success! Tuned the secrets engine at: secret/
```

or via the API:

```
$ cat payload.json
{
  "options": {
      "version": "2"
  }
}
```

```
$ curl \
    --header "X-Vault-Token: ..." \
    --request POST \
    --data @payload.json \
    http://127.0.0.1:8200/v1/sys/mounts/secret/tune
```

ACL rules

The version 2 kv store uses a prefixed API, which is different from the version 1 API. Before upgrading from a version 1 kv the ACL rules should be changed. Also different paths in the version 2 API can be ACL'ed differently.

Writing and reading versions are prefixed with the `data/` path. This policy that worked for the version 1 kv:

```
path "secret/dev/team-1/*" {
  capabilities = ["create", "update", "read"]
}
```

Should be changed to:

```
path "secret/data/dev/team-1/*" {
  capabilities = ["create", "update", "read"]
}
```

There are different levels of data deletion for this backend. To grant a policy the permissions to delete the latest version of a key:

```
path "secret/data/dev/team-1/*" {
  capabilities = ["delete"]
}
```

To allow the policy to delete any version of a key:

```
path "secret/delete/dev/team-1/*" {
  capabilities = ["update"]
}
```

To allow a policy to undelete data:

```
path "secret/undelete/dev/team-1/*" {
  capabilities = ["update"]
}
```

To allow a policy to destroy versions:

```
path "secret/destroy/dev/team-1/*" {
  capabilities = ["update"]
}
```

To allow a policy to list keys:

```
path "secret/metadata/dev/team-1/*" {
  capabilities = ["list"]
}
```

To allow a policy to view metadata for each version:

```
path "secret/metadata/dev/team-1/*" {
  capabilities = ["read"]
}
```

To allow a policy to permanently remove all versions and metadata for a key:

```
path "secret/metadata/dev/team-1/*" {
  capabilities = ["delete"]
}
```

The `allowed_parameters`, `denied_parameters`, and `required_parameters` fields are not supported for policies used with the version 2 kv store. See the Policies Concepts for a description of these parameters.

Usage

After the secrets engine is configured and a user/machine has an Stronghold token with the proper permission, it can generate credentials. The `kv` secrets engine allows for writing keys with arbitrary values.

The path-like KV-v1 syntax for referencing a secret (`secret/foo`) can still be used in KV-v2, but we recommend using the `-mount=secret` flag syntax to avoid mistaking it for the actual path to the secret (`secret/data/foo` is the real path).

Writing/Reading arbitrary data

1. Write arbitrary data:

```
$ d8 stronghold kv put -mount=secret my-secret foo=a bar=b
Key              Value
---              -----
created_time     2024-06-19T17:20:22.985303Z
custom_metadata  <nil>
deletion_time    n/a
destroyed        false
version          1
```

2. Read arbitrary data:

```
$ d8 stronghold kv get -mount=secret my-secret
====== Metadata ======
Key              Value
---              -----
created_time     2024-06-19T17:20:22.985303Z
custom_metadata  <nil>
deletion_time    n/a
destroyed        false
version          1

====== Data ======
Key         Value
---         -----
foo         a
bar         b
```

3. Write another version, the previous version will still be accessible. The `-cas` flag can optionally be passed to perform a check-and-set operation. If not set the write will be allowed. In order for a write to be successful, `cas` must be set to the current version of the secret. If set to 0 a write will only be allowed if the key doesn’t exist as unset keys do not have any version information. Also remember that soft deletes do not remove any underlying version data from storage. In order to write to a soft deleted key, the cas parameter must match the key’s current version.

```
$ d8 stronghold kv put -mount=secret -cas=1 my-secret foo=aa bar=bb
Key              Value
---              -----
created_time     2024-06-19T17:22:23.369372Z
custom_metadata  <nil>
deletion_time    n/a
destroyed        false
version          2
```

4. Reading now will return the newest version of the data:

```
$ d8 stronghold kv get -mount=secret my-secret
====== Metadata ======
Key              Value
---              -----
created_time     2024-06-19T17:22:23.369372Z
custom_metadata  <nil>
deletion_time    n/a
destroyed        false
version          2

====== Data ======
Key         Value
---         -----
foo         aa
bar         bb
```

5. Partial updates can be accomplished using the `d8 stronghold kv patch` command. A command will initially attempt an HTTP `PATCH` request which requires the `patch` ACL capability. The `PATCH` request will fail if the token used is associated with a policy that does not contain the `patch` capability. In this case the command will perform a read, local update, and subsequent write which require both the `read` and `update` ACL capabilities.

   The `-cas` flag can optionally be passed to perform a check-and-set operation. It will only be used in the case of the initial `PATCH` request. The read-then-write flow will use the `version` value from the secret returned by the read to perform a check-and-set operation in the subsequent write.

```
$ d8 stronghold kv patch -mount=secret -cas=2 my-secret bar=bbb
Key              Value
---              -----
created_time     2024-06-19T17:23:49.199802Z
custom_metadata  <nil>
deletion_time    n/a
destroyed        false
version          3
```

6. The `d8 stronghold kv patch` command also supports a `-method` flag which can be used to specify HTTP `PATCH` or read-then-write. The supported values are `patch` and `rw` for HTTP `PATCH` and read-then-write, respectively.

   Perform a patch using the `patch` method:

   ```
   $ d8 stronghold kv patch -mount=secret -method=patch -cas=2 my-secret bar=bbb
   Key              Value
   ---              -----
   created_time     2024-06-19T17:23:49.199802Z
   custom_metadata  <nil>
   deletion_time    n/a
   destroyed        false
   version          3
   ```

   Perform a patch using the read-then-write method:

   ```
   $ d8 stronghold kv patch -mount=secret -method=rw my-secret bar=bbb
   Key              Value
   ---              -----
   created_time     2024-06-19T17:23:49.199802Z
   custom_metadata  <nil>
   deletion_time    n/a
   destroyed        false
   version          3
   ```

7. Reading after a patch will return the newest version of the data in which only the specified fields were updated:

   ```
   $ d8 stronghold kv get -mount=secret my-secret
   ====== Metadata ======
   Key              Value
   ---              -----
   created_time     2024-06-19T17:23:49.199802Z
   custom_metadata  <nil>
   deletion_time    n/a
   destroyed        false
   version          3

   ====== Data ======
   Key         Value
   ---         -----
   foo         aa
   bar         bbb
   ```

8. Previous versions can be accessed with the `-version` flag:

```
$ d8 stronghold kv get -mount=secret -version=1 my-secret
====== Metadata ======
Key              Value
---              -----
created_time     2024-06-19T17:20:22.985303Z
custom_metadata  <nil>
deletion_time    n/a
destroyed        false
version          1

====== Data ======
Key         Value
---         -----
foo         a
bar         b
```

You can also use Stronghold's password policy feature to generate arbitrary values.

1. Write a password policy:

```
$ d8 stronghold write sys/policies/password/example policy=-<<EOF

  length=20

  rule "charset" {
    charset = "abcdefghij0123456789"
    min-chars = 1
  }

  rule "charset" {
    charset = "!@#$%^&*STUVWXYZ"
    min-chars = 1
  }

EOF
```

2. Write data using the `example` policy:

```
$ d8 stronghold kv put -mount=secret my-generated-secret \
    password=$(d8 stronghold read -field password sys/policies/password/example/
generate)
```

Example output:

```
========= Secret Path =========
secret/data/my-generated-secret

======= Metadata =======
Key                Value
---                -----
created_time       2023-05-10T14:32:32.37354939Z
custom_metadata    <nil>
deletion_time      n/a
destroyed          false
version            1
```

3. Read the generated data:

```
$ d8 stronghold kv get -mount=secret my-generated-secret
========= Secret Path =========
secret/data/my-generated-secret

======= Metadata =======
Key                Value
---                -----
created_time       2023-05-10T14:32:32.37354939Z
custom_metadata    <nil>
deletion_time      n/a
destroyed          false
version            1

====== Data ======
Key         Value
---         -----
password    !hh&be1e4j16dVc0ggae
```

Deleting and destroying data

When deleting data the standard `d8 stronghold kv delete` command will perform a soft delete. It will mark the version as deleted and populate a `deletion_time` timestamp. Soft deletes do not remove the underlying version data from storage, which allows the version to be undeleted. The `d8 stronghold kv undelete` command handles undeleting versions.

A version's data is permanently deleted only when the key has more versions than are allowed by the max-versions setting, or when using `d8 stronghold kv destroy`. When the destroy command is used the underlying version data will be removed and the key metadata will be marked as destroyed. If a version is cleaned up by going over max-versions the version metadata will also be removed from the key.

See the commands below for more information:

1. The latest version of a key can be deleted with the delete command, this also takes a `-versions` flag to delete prior versions:

```
$ d8 stronghold kv delete -mount=secret my-secret
Success! Data deleted (if it existed) at: secret/data/my-secret
```

2. Versions can be undeleted:

```
$ d8 stronghold kv undelete -mount=secret -versions=2 my-secret
Success! Data written to: secret/undelete/my-secret

$ d8 stronghold kv get -mount=secret my-secret
====== Metadata ======
Key              Value
---              -----
created_time     2024-06-19T17:23:21.834403Z
custom_metadata  <nil>
deletion_time    n/a
destroyed        false
version          2

====== Data ======
Key         Value
---         -----
my-value    short-lived-s3cr3t
```

3. Destroying a version permanently deletes the underlying data:

```
$ d8 stronghold kv destroy -mount=secret -versions=2 my-secret
Success! Data written to: secret/destroy/my-secret
```

Key metadata

All versions and key metadata can be tracked with the metadata command & API. Deleting the metadata key will cause all metadata and versions for that key to be permanently removed.

See the commands below for more information:

1. All metadata and versions for a key can be viewed:

```
$ d8 stronghold kv metadata get -mount=secret my-secret
========== Metadata ==========
Key                     Value
---                     -----
cas_required            false
created_time            2024-06-19T17:20:22.985303Z
current_version         2
custom_metadata         <nil>
delete_version_after    0s
max_versions            0
oldest_version          0
updated_time            2024-06-19T17:22:23.369372Z

====== Version 1 ======
Key              Value
---              -----
created_time     2024-06-19T17:20:22.985303Z
deletion_time    n/a
destroyed        false

====== Version 2 ======
Key              Value
---              -----
created_time     2024-06-19T17:22:23.369372Z
deletion_time    n/a
destroyed        true
```

2. The metadata settings for a key can be configured:

```
$ d8 stronghold kv metadata put -mount=secret -max-versions 2 -delete-version-
after="3h25m19s" my-secret
Success! Data written to: secret/metadata/my-secret
```

Delete-version-after settings will apply only to new versions. Max versions changes will be applied on next write:

```
$ d8 stronghold kv put -mount=secret my-secret my-value=newer-s3cr3t
Key              Value
---              -----
created_time     2024-06-19T17:31:16.662563Z
custom_metadata  <nil>
deletion_time    2024-06-19T20:56:35.662563Z
destroyed        false
version          4
```

Once a key has more versions than max versions the oldest versions are cleaned up:

```
$ d8 stronghold kv metadata get -mount=secret my-secret
========== Metadata ==========
Key                     Value
---                     -----
cas_required            false
created_time            2024-06-19T17:20:22.985303Z
current_version         4
custom_metadata         <nil>
delete_version_after    3h25m19s
max_versions            2
oldest_version          3
updated_time            2024-06-19T17:31:16.662563Z

====== Version 3 ======
Key              Value
---              -----
created_time     2024-06-19T17:23:21.834403Z
deletion_time    n/a
destroyed        true

====== Version 4 ======
Key              Value
---              -----
created_time     2024-06-19T17:31:16.662563Z
deletion_time    2024-06-19T20:56:35.662563Z
destroyed        false
```

A secret's key metadata can contain custom metadata used to describe the secret. The data will be stored as string-to-string key-value pairs. The `-custom-metadata` flag can be repeated to add multiple key-value pairs.

The `d8 stronghold kv metadata put` command can be used to fully overwrite the value of `custom_metadata`:

```
$ d8 stronghold kv metadata put -mount=secret -custom-metadata=foo=abc -custom-
metadata=bar=123 my-secret
Success! Data written to: secret/metadata/my-secret

$ d8 stronghold kv get -mount=secret my-secret
====== Metadata ======
Key              Value
---              -----
created_time     2024-06-19T17:22:23.369372Z
custom_metadata  map[bar:123 foo:abc]
deletion_time    n/a
destroyed        false
version          2

====== Data ======
Key         Value
---         -----
foo         aa
bar         bb
```

The `d8 stronghold kv metadata patch` command can be used to partially overwrite the value of `custom_metadata`. The following invocation will update `custom_metadata` sub-field `foo` but leave `bar` untouched:

```
$ d8 stronghold kv metadata patch -mount=secret -custom-metadata=foo=def my-
secret
Success! Data written to: secret/metadata/my-secret
```

```
$ d8 stronghold kv get -mount=secret my-secret
====== Metadata ======
Key              Value
---              -----
created_time     2024-06-19T17:22:23.369372Z
custom_metadata  map[bar:123 foo:def]
deletion_time    n/a
destroyed        false
version          2

====== Data ======
Key         Value
---         -----
foo         aa
bar         bb
```

3. Permanently delete all metadata and versions for a key:

```
$ d8 stronghold kv metadata delete -mount=secret my-secret
Success! Data deleted (if it existed) at: secret/metadata/my-secret
```

KV1/KV2 replication

In development

PKI Secrets engine

This document provides a brief overview of setting up an Stronghold PKI Secrets Engine with a Root CA certificate.

Mount the backend

The first step to using the PKI backend is to mount it. Unlike the `kv` backend, the `pki` backend is not mounted by default.

```
$ d8 stronghold secrets enable pki
Successfully mounted 'pki' at 'pki'!
```

Configure a CA certificate

Next, Stronghold must be configured with a CA certificate and associated private key. We'll take advantage of the backend's self-signed root generation support, but Stronghold also supports generating an intermediate CA (with a CSR for signing) or setting a PEM-encoded certificate and private key bundle directly into the backend.

Generally you'll want a root certificate to only be used to sign CA intermediate certificates, but for this example we'll proceed as if you will issue certificates directly from the root. As it's a root, we'll want to set a long maximum life time for the certificate; since it honors the maximum mount TTL, first we adjust that:

```
$ d8 stronghold secrets tune -max-lease-ttl=87600h pki
Successfully tuned mount 'pki'!
```

That sets the maximum TTL for secrets issued from the mount to 10 years. (Note that roles can further restrict the maximum TTL.)

Now, we generate our root certificate:

```
$ d8 stronghold write pki/root/generate/internal common_name=myopenbao.com ttl=87600h
Key             Value
---             -----
certificate     -----BEGIN CERTIFICATE-----
MIIDNTCCAh2gAwIBAgIUJqrw/9EDZbp4DExaLjh0vSAHyBgwDQYJKoZIhvcNAQEL
BQAwFjEUMBIGA1UEAxMLbXl2YXVsdC5jb20wHhcNMTcxMjA4MTkyMzIwWhcNMjcx
MjA2MTkyMzQ5WjAWMRQwEgYDVQQDEwtteXZhdWx0LmNvbTCCASIwDQYJKoZIhvcN
AQEBBQADggEPADCCAQoCggEBAKY/vJ6sRFym+yFYUneoVtDmOCaDKAQiGzQw0IXL
wgMBBb82iKpYj5aQjXZGIl+VkVnCi+M2AQ/iYXWZf1kTAdle4A6OC4+VefSIa2b4
eB7R8aiGTce62jB95+s5/YgrfIqk6igfpCSXYLE8ubNDA2/+cqvjhku1UzlvKBX2
hIlgWkKlrsnybHN+B/3Usw9Km/87rzoDR3OMxLV55YPHiq6+olIfSSwKAPjH8LZm
uM1ITLG3WQUl8ARF17Dj+wOKqbUG38PduVwKL5+qPksrvNwlmCP7Kmjncc6xnYp6
5lfr7V4DC/UezrJYCIb0g/SvtxoN1OuqmmvSTKiEE7hVOAcCAwEAAaN7MHkwDgYD
VR0PAQH/BAQDAgEGMA8GA1UdEwEB/wQFMAMBAf8wHQYDVR0OBBYEFECKdYM4gDbM
kxRZA2wR4f/yNhQUMB8GA1UdIwQYMBaAFECKdYM4gDbMkxRZA2wR4f/yNhQUMBYG
A1UdEQQPMA2CC215dmF1bHQuY29tMA0GCSqGSIb3DQEBCwUAA4IBAQCCJKZPcjjn
7mvD2+sr6lx4DW/vJwVSW8eTuLtOLNu6/aFhcgTY/OOB8q4n6iHuLrEt8/RV7RJI
obRx74SfK9BcOLt4+DHGnFXqu2FNVnhDMOKarj41yGyXlJaQRUPYf6WJJLF+ZphN
nNsZqHJHBfZtpJpE5Vywx3pah08B5yZHk1ItRPEz7EY3uwBI/CJoBb+P5Ahk6krc
LZ62kFwstkVuFp43o3K7cRNexCIsZGx2tsyZ0nyqDUFsBr66xwUfn3C+/1CDc9YL
zjq+8nI2ooIrj4ZKZCOm2fKd1KeGN/CZD7Ob6uNhXrd0Tjwv00a7nffvYQkl/1V5
BT55jevSPVVu
-----END CERTIFICATE-----
expiration      1828121029
issuing_ca      -----BEGIN CERTIFICATE-----
MIIDNTCCAh2gAwIBAgIUJqrw/9EDZbp4DExaLjh0vSAHyBgwDQYJKoZIhvcNAQEL
BQAwFjEUMBIGA1UEAxMLbXl2YXVsdC5jb20wHhcNMTcxMjA4MTkyMzIwWhcNMjcx
MjA2MTkyMzQ5WjAWMRQwEgYDVQQDEwtteXZhdWx0LmNvbTCCASIwDQYJKoZIhvcN
AQEBBQADggEPADCCAQoCggEBAKY/vJ6sRFym+yFYUneoVtDmOCaDKAQiGzQw0IXL
wgMBBb82iKpYj5aQjXZGIl+VkVnCi+M2AQ/iYXWZf1kTAdle4A6OC4+VefSIa2b4
eB7R8aiGTce62jB95+s5/YgrfIqk6igfpCSXYLE8ubNDA2/+cqvjhku1UzlvKBX2
hIlgWkKlrsnybHN+B/3Usw9Km/87rzoDR3OMxLV55YPHiq6+olIfSSwKAPjH8LZm
uM1ITLG3WQUl8ARF17Dj+wOKqbUG38PduVwKL5+qPksrvNwlmCP7Kmjncc6xnYp6
5lfr7V4DC/UezrJYCIb0g/SvtxoN1OuqmmvSTKiEE7hVOAcCAwEAAaN7MHkwDgYD
VR0PAQH/BAQDAgEGMA8GA1UdEwEB/wQFMAMBAf8wHQYDVR0OBBYEFECKdYM4gDbM
kxRZA2wR4f/yNhQUMB8GA1UdIwQYMBaAFECKdYM4gDbMkxRZA2wR4f/yNhQUMBYG
A1UdEQQPMA2CC215dmF1bHQuY29tMA0GCSqGSIb3DQEBCwUAA4IBAQCCJKZPcjjn
7mvD2+sr6lx4DW/vJwVSW8eTuLtOLNu6/aFhcgTY/OOB8q4n6iHuLrEt8/RV7RJI
obRx74SfK9BcOLt4+DHGnFXqu2FNVnhDMOKarj41yGyXlJaQRUPYf6WJJLF+ZphN
nNsZqHJHBfZtpJpE5Vywx3pah08B5yZHk1ItRPEz7EY3uwBI/CJoBb+P5Ahk6krc
LZ62kFwstkVuFp43o3K7cRNexCIsZGx2tsyZ0nyqDUFsBr66xwUfn3C+/1CDc9YL
zjq+8nI2ooIrj4ZKZCOm2fKd1KeGN/CZD7Ob6uNhXrd0Tjwv00a7nffvYQkl/1V5
BT55jevSPVVu
-----END CERTIFICATE-----
serial_number   26:aa:f0:ff:d1:03:65:ba:78:0c:4c:5a:2e:38:74:bd:20:07:c8:18
```

The returned certificate is purely informational; it and its private key are safely stored in the backend mount.

Set URL configuration

Generated certificates can have the CRL location and the location of the issuing certificate encoded. These values must be set manually and typically to FQDN associated to the Stronghold server, but can be changed at any time.

```
$ d8 stronghold write pki/config/urls issuing_certificates="http://
openbao.example.com:8200/v1/pki/ca" crl_distribution_points="http://
openbao.example.com:8200/v1/pki/crl"
Success! Data written to: pki/ca/urls
```

Configure a role

The next step is to configure a role. A role is a logical name that maps to a policy used to generate those credentials. For example, let’s create an “example-dot-com” role:

```
$ d8 stronghold write pki/roles/example-dot-com \
    allowed_domains=example.com \
    allow_subdomains=true max_ttl=72h
Success! Data written to: pki/roles/example-dot-com
```

Issue certificates

By writing to the `roles/example-dot-com` path we are defining the `example-dot-com` role. To generate a new certificate, we simply write to the `issue` endpoint with that role name: Stronghold is now configured to create and manage certificates!

```
$ d8 stronghold write pki/issue/example-dot-com \
    common_name=blah.example.com
Key                 Value
---                 -----
certificate         -----BEGIN CERTIFICATE-----
MIIDvzCCAqegAwIBAgIUWQuvpMpA2ym36EoiYyf3Os5UeIowDQYJKoZIhvcNAQEL
BQAwFjEUMBIGA1UEAxMLbXl2YXVsdC5jb20wHhcNMTcxMjA4MTkyNDA1WhcNMTcx
MjExMTkyNDM1WjAbMRkwFwYDVQQDExBibGFoLmV4YW1wbGUuY29tMIIBIjANBgkq
hkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEA1CU93lVgcLXGPxRGTRT3GM5wqytCo7Z6
gjfoHyKoPCAqjRdjsYgp1FMvumNQKjUat5KTtr2fypbOnAURDCh4bN/omcj7eAqt
ldJ8mf8CtKUaaJ1kp3R6RRFY/u96BnmKUG8G7oDeEDsKlXuEuRcNbGlGF8DaM/O1
HFa57cM/8yFB26Nj5wBoG5Om6ee5+W+14Qee8AB6OJbsf883Z+zvhJTaB0QM4ZUq
uAMoMVEutWhdI5EFm5OjtMeMu2U+iJl2XqqgQ/JmLRjRdMn1qd9TzTaVSnjoZ97s
jHK444Px1m45einLqKUJ+Ia2ljXYkkItJj9Ut6ZSAP9fHlAtX84W3QIDAQABo4H/
MIH8MA4GA1UdDwEB/wQEAwIDqDAdBgNVHSUEFjAUBggrBgEFBQcDAQYIKwYBBQUH
AwIwHQYDVR0OBBYEFH/YdObW6T94U0zuU5hBfTfU5pt1MB8GA1UdIwQYMBaAFECK
dYM4gDbMkxRZA2wR4f/yNhQUMDsGCCsGAQUFBwEBBC8wLTArBggrBgEFBQcwAoYf
aHR0cDovLzEyNy4wLjAuMTo4MjAwL3YxL3BraS9jYTAbBgNVHREEFDASghBibGFo
LmV4YW1wbGUuY29tMDEGA1UdHwQqMCgwJqAkoCKGIGh0dHA6Ly8xMjcuMC4wLjE6
ODIwMC92MS9wa2kvY3JsMA0GCSqGSIb3DQEBCwUAA4IBAQCDXbHV68VayweB2tkb
KDdCaveaTULjCeJUnm9UT/6C0YqC/RxTAjdKFrilK49elOA3rAtEL6dmsDP2yH25
ptqi2iU+y99HhZgu0zkS/p8elYN3+l+0O7pOxayYXBkFf5t0TlEWSTb7cW+Etz/c
MvSqx6vVvspSjB0PsA3eBq0caZnUJv2u/TEiUe7PPY0UmrZxp/R/P/kE54yI3nWN
4Cwto6yUwScOPbVR1d3hE2KU2toiVkEoOk17UyXWTokbG8rG0KLj99zu7my+Fyre
sjV5nWGDSMZODEsGxHOC+JgNAC1z3n14/InFNOsHICnA5AnJzQdSQQjvcZHN2NyW
+t4f
-----END CERTIFICATE-----
issuing_ca          -----BEGIN CERTIFICATE-----
MIIDNTCCAh2gAwIBAgIUJqrw/9EDZbp4DExaLjh0vSAHyBgwDQYJKoZIhvcNAQEL
BQAwFjEUMBIGA1UEAxMLbXl2YXVsdC5jb20wHhcNMTcxMjA4MTkyMzIwWhcNMjcx
MjA2MTkyMzQ5WjAWMRQwEgYDVQQDEwtteXZhdWx0LmNvbTCCASIwDQYJKoZIhvcN
AQEBBQADggEPADCCAQoCggEBAKY/vJ6sRFym+yFYUneoVtDmOCaDKAQiGzQw0IXL
wgMBBb82iKpYj5aQjXZGIl+VkVnCi+M2AQ/iYXWZf1kTAdle4A6OC4+VefSIa2b4
eB7R8aiGTce62jB95+s5/YgrfIqk6igfpCSXYLE8ubNDA2/+cqvjhku1UzlvKBX2
hIlgWkKlrsnybHN+B/3Usw9Km/87rzoDR3OMxLV55YPHiq6+olIfSSwKAPjH8LZm
uM1ITLG3WQUl8ARF17Dj+wOKqbUG38PduVwKL5+qPksrvNwlmCP7Kmjncc6xnYp6
5lfr7V4DC/UezrJYCIb0g/SvtxoN1OuqmmvSTKiEE7hVOAcCAwEAAaN7MHkwDgYD
VR0PAQH/BAQDAgEGMA8GA1UdEwEB/wQFMAMBAf8wHQYDVR0OBBYEFECKdYM4gDbM
kxRZA2wR4f/yNhQUMB8GA1UdIwQYMBaAFECKdYM4gDbMkxRZA2wR4f/yNhQUMBYG
A1UdEQQPMA2CC215dmF1bHQuY29tMA0GCSqGSIb3DQEBCwUAA4IBAQCCJKZPcjjn
7mvD2+sr6lx4DW/vJwVSW8eTuLtOLNu6/aFhcgTY/OOB8q4n6iHuLrEt8/RV7RJI
obRx74SfK9BcOLt4+DHGnFXqu2FNVnhDMOKarj41yGyXlJaQRUPYf6WJJLF+ZphN
nNsZqHJHBfZtpJpE5Vywx3pah08B5yZHk1ItRPEz7EY3uwBI/CJoBb+P5Ahk6krc
LZ62kFwstkVuFp43o3K7cRNexCIsZGx2tsyZ0nyqDUFsBr66xwUfn3C+/1CDc9YL
zjq+8nI2ooIrj4ZKZCOm2fKd1KeGN/CZD7Ob6uNhXrd0Tjwv00a7nffvYQkl/1V5
BT55jevSPVVu
-----END CERTIFICATE-----
private_key         -----BEGIN RSA PRIVATE KEY-----
```

```
MIIEpAIBAAKCAQEA1CU93lVgcLXGPxRGTRT3GM5wqytCo7Z6gjfoHyKoPCAqjRdj
sYgp1FMvumNQKjUat5KTtr2fypbOnAURDCh4bN/omcj7eAqtldJ8mf8CtKUaaJ1k
p3R6RRFY/u96BnmKUG8G7oDeEDsKlXuEuRcNbGlGF8DaM/O1HFa57cM/8yFB26Nj
5wBoG5Om6ee5+W+14Qee8AB6OJbsf883Z+zvhJTaB0QM4ZUquAMoMVEutWhdI5EF
m5OjtMeMu2U+iJl2XqqgQ/JmLRjRdMn1qd9TzTaVSnjoZ97sjHK444Px1m45einL
qKUJ+Ia2ljXYkkItJj9Ut6ZSAP9fHlAtX84W3QIDAQABAoIBAQCf5YIANfF+gkNt
/+YM6yRi+hZJrU2I/1zPETxPW1vaFZR8y4hEoxCEDD8JCRm+9k+w1TWoorvxgkEv
r1HuDALYbNtwLd/71nCHYCKyH1b2uQpyl07qOAyASlb9r5oVjz4E6eobkd3N9fJA
QN0EdK+VarN968mLJsD3Hxb8chGdObBCQ+LO+zdqQLaz+JwhfnK98rm6huQtYK3w
ccd0OwoVmtZz2eJl11TJkB9fi4WqJyxl4wST7QC80LstB1deR78oDmN5WUKU12+G
4Mrgc1hRwUSm18HTTgAhaA4A3rjPyirBohb5Sf+jJxusnnay7tvWeMnIiRI9mqCE
dr3tLrcxAoGBAPL+jHVUF6sxBqm6RTe8Ewg/8RrGmd69oB71QlVUrLYyC96E2s56
19dcyt5U2z+F0u9wlwR1rMb2BJIXbxlNk+i87IHmpOjCMS38SPZYWLHKj02eGfvA
MjKKqEjNY/md9eVAVZIWSEy63c4UcBK1qUH3/5PNlyjk53gCOI/4OXX/AoGBAN+A
Alyd6A/pyHWq8WMyAlV18LnzX8XktJ07xrNmjbPGD5sEHp+Q9V33NitOZpu3bQL+
gCNmcrodrbr9LBV83bkAOVJrf82SPaBesV+ATY7ZiWpqvHTmcoS7nglM2XTr+uWR
Y9JGdpCE9U5QwTc6qfcn7Eqj7yNvvHMrT+1SHwsjAoGBALQyQEbhzYuOF7rV/26N
ci+z+0A39vNO++b5Se+tk0apZlPlgb2NK3LxxR+LHevFed9GRzdvbGk/F7Se3CyP
cxgswdazC6fwGjhX1mOYsG1oIU0V6X7f0FnaqWETrwf1M9yGEO78xzDfgozIazP0
s0fQeR9KXsZcuaotO3TIRxRRAoGAMFIDsLRvDKm1rkL0B0czm/hwwDMu/KDyr5/R
2M2OS1TB4PjmCgeUFOmyq3A63OWuStxtJboribOK8Qd1dXvWj/3NZtVY/z/j1P1E
Ceq6We0MOZa0Ae4kyi+p/kbAKPgv+VwSoc6cKailRHZPH7quLoJSIt0IgbfRnXC6
ygtcLNMCgYBwiPw2mTYvXDrAcO17NhK/r7IL7BEdFdx/w8vNJQp+Ub4OO3Iw6ARI
vXxu6A+Qp50jra3UUtnI+hIirMS+XEeWqJghK1js3ZR6wA/ZkYZw5X1RYuPexb/4
6befxmnEuGSbsgvGqYYTf5Z0vgsw4tAHfNS7TqSulYH06CjeG1F8DQ==
-----END RSA PRIVATE KEY-----
private_key_type    rsa
serial_number       59:0b:af:a4:ca:40:db:29:b7:e8:4a:22:63:27:f7:3a:ce:54:78:8a
```

Stronghold has now generated a new set of credentials using the `example-dot-com` role configuration. Here we see the dynamically generated private key and certificate.

Using ACLs, it is possible to restrict using the pki backend such that trusted operators can manage the role definitions, and both users and applications are restricted in the credentials they are allowed to read.

If you get stuck at any time, simply run `d8 stronghold path-help pki` or with a subpath for interactive help output.

Cubbyhole Secrets engine

The `cubbyhole` secrets engine is used to store arbitrary secrets within the configured physical storage for Stronghold namespaced to a token. In `cubbyhole`, paths are scoped per token. No token can access another token's cubbyhole. When the token expires, its cubbyhole is destroyed.

Also unlike the `kv` secrets engine, because the cubbyhole's lifetime is linked to that of an authentication token, there is no concept of a TTL or refresh interval for values contained in the token's cubbyhole.

Writing to a key in the `cubbyhole` secrets engine will completely replace the old value.

Setup

Most secrets engines must be configured in advance before they can perform their functions. These steps are usually completed by an operator or configuration management tool.

The `cubbyhole` secrets engine is enabled by default. It cannot be disabled, moved, or enabled multiple times.

Usage

After the secrets engine is configured and a user/machine has an Stronghold token with the proper permission, it can generate credentials. The `cubbyhole` secrets engine allows for writing keys with arbitrary values.

1. Write arbitrary data:

```
$ d8 stronghold write cubbyhole/my-secret my-value=s3cr3t
Success! Data written to: cubbyhole/my-secret
```

2. Read arbitrary data:

```
$ d8 stronghold read cubbyhole/my-secret
Key         Value
---         -----
my-value    s3cr3t
```

SSH Secrets engine

The signed SSH certificates is the simplest and most powerful in terms of setup complexity and in terms of being platform agnostic. By leveraging Stronghold's powerful CA capabilities and functionality built into OpenSSH, clients can SSH into target hosts using their own local SSH keys.

In this section, the term "**client**" refers to the person or machine performing the SSH operation. The "**host**" refers to the target machine. If this is confusing, substitute "client" with "user".

This page will show a quick start for this secrets engine. For detailed documentation on every path, use `bao path-help` after mounting the secrets engine.

Client key signing

Before a client can request their SSH key be signed, the Stronghold SSH secrets engine must be configured. Usually an Stronghold administrator or security team performs these steps. It is also possible to automate these actions using a configuration management tool like Chef, Puppet, Ansible, or Salt.

Signing key & role configuration

The following steps are performed in advance by an Stronghold administrator, security team, or configuration management tooling.

1. Mount the secrets engine. Like all secrets engines in Stronghold, the SSH secrets engine must be mounted before use.

   ```
   $ d8 stronghold secrets enable -path=ssh-client-signer ssh
   Successfully mounted 'ssh' at 'ssh-client-signer'!
   ```

   This enables the SSH secrets engine at the path “ssh-client-signer”. It is possible to mount the same secrets engine multiple times using different `-path` arguments. The name “ssh-client-signer” is not special - it can be any name, but this documentation will assume “ssh-client-signer”.

2. Configure Stronghold with a CA for signing client keys using the `/config/ca` endpoint. If you do not have an internal CA, Stronghold can generate a keypair for you.

   ```
   $ d8 stronghold write ssh-client-signer/config/ca generate_signing_key=true
   Key             Value
   ---             -----
   public_key      ssh-rsa AAAAB3NzaC1yc2EA...
   ```

   If you already have a keypair, specify the public and private key parts as part of the payload:

   ```
   $ d8 stronghold write ssh-client-signer/config/ca \
       private_key="..." \
       public_key="..."
   ```

   The SSH secrets engine allows multiple Certificate Authority (CA) certificates (“issuers”) to be configured in a single mount. This feature is designed to facilitate CA rotation. When configuring a CA, one issuer is designated as the default - its operations will be used whenever no specific issuer is referenced during role creation. The default issuer can be changed at any time by either generating a new CA or updating it through the configuration endpoint, enabling seamless CA rotation.

   Regardless of whether it is generated or uploaded, the client signer public key is accessible via the API at the `/public_key` endpoint or the CLI (see next step).

3. Add the public key to all target host’s SSH configuration. This process can be manual or automated using a configuration management tool. The public key is accessible via the API and does not require authentication.

   ```
   curl -o /etc/ssh/trusted-user-ca-keys.pem http://127.0.0.1:8200/v1/ssh-client-signer/public_key
   ```

```
d8 stronghold read -field=public_key ssh-client-signer/config/ca > /etc/ssh/
trusted-user-ca-keys.pem
```

Add the path where the public key contents are stored to the SSH configuration file as the `TrustedUserCAKeys` option.

```
# /etc/ssh/sshd_config
# ...
TrustedUserCAKeys /etc/ssh/trusted-user-ca-keys.pem
```

Restart the SSH service to pick up the changes.

4. Create a named Stronghold role for signing client keys.

   Because of the way some SSH certificate features are implemented, options are passed as a map. The following example adds the `permit-pty` extension to the certificate, and allows the user to specify their own values for `permit-pty` and `permit-port-forwarding` when requesting the certificate.

```
$ d8 stronghold write ssh-client-signer/roles/my-role -<<"EOH"
{
  "algorithm_signer": "rsa-sha2-256",
  "allow_user_certificates": true,
  "allowed_users": "*",
  "allowed_extensions": "permit-pty,permit-port-forwarding",
  "default_extensions": {
    "permit-pty": ""
  },
  "key_type": "ca",
  "default_user": "ubuntu",
  "ttl": "30m0s"
}
EOH
```

Client SSH authentication

The following steps are performed by the client (user) that wants to authenticate to machines managed by Stronghold. These commands are usually run from the client's local workstation.

1. Locate or generate the SSH public key. Usually this is `~/.ssh/id_rsa.pub`. If you do not have an SSH keypair, generate one:

```
ssh-keygen -t rsa -C "user@example.com"
```

2. Ask Stronghold to sign your **public key**. This file usually ends in `.pub` and the contents begin with `ssh-rsa ...`.

```
$ d8 stronghold write ssh-client-signer/sign/my-role \
    public_key=@$HOME/.ssh/id_rsa.pub

Key             Value
---             -----
serial_number   c73f26d2340276aa
signed_key      ssh-rsa-cert-v01@openssh.com AAAAHHNzaC1...
```

The result will include the serial and the signed key. This signed key is another public key.

To customize the signing options, use a JSON payload:

```
$ d8 stronghold write ssh-client-signer/sign/my-role -<<"EOH"
{
  "public_key": "ssh-rsa AAA...",
  "valid_principals": "my-user",
  "key_id": "custom-prefix",
  "extensions": {
    "permit-pty": "",
    "permit-port-forwarding": ""
  }
}
EOH
```

3. Save the resulting signed, public key to disk. Limit permissions as needed.

```
$ d8 stronghold write -field=signed_key ssh-client-signer/sign/my-role \
    public_key=@$HOME/.ssh/id_rsa.pub > signed-cert.pub
```

If you are saving the certificate directly beside your SSH keypair, suffix the name with `-cert.pub` (`~/.ssh/id_rsa-cert.pub`). With this naming scheme, OpenSSH will automatically use it during authentication.

4. (Optional) View enabled extensions, principals, and metadata of the signed key.

```
ssh-keygen -Lf ~/.ssh/signed-cert.pub
```

5. SSH into the host machine using the signed key. You must supply both the signed public key from Stronghold **and** the corresponding private key as authentication to the SSH call.

```
ssh -i signed-cert.pub -i ~/.ssh/id_rsa username@10.0.23.5
```

Host key signing

For an added layer of security, we recommend enabling host key signing. This is used in conjunction with client key signing to provide an additional integrity layer. When enabled, the SSH agent will verify the target host is valid and trusted before attempting to SSH. This will reduce the probability of a user accidentally SSHing into an unmanaged or malicious machine.

Signing key configuration

1. Mount the secrets engine. For the most security, mount at a different path from the client signer.

   ```
   $ d8 stronghold secrets enable -path=ssh-host-signer ssh
   Successfully mounted 'ssh' at 'ssh-host-signer'!
   ```

2. Configure Stronghold with a CA for signing host keys using the `/config/ca` endpoint. If you do not have an internal CA, Stronghold can generate a keypair for you.

   ```
   $ d8 stronghold write ssh-host-signer/config/ca generate_signing_key=true
   Key             Value
   ---             -----
   public_key      ssh-rsa AAAAB3NzaC1yc2EA...
   ```

   If you already have a keypair, specify the public and private key parts as part of the payload:

   ```
   $ d8 stronghold write ssh-host-signer/config/ca \
       private_key="..." \
       public_key="..."
   ```

   Regardless of whether it is generated or uploaded, the host signer public key is accessible via the API at the `/public_key` endpoint.

3. Extend host key certificate TTLs.

   ```
   d8 stronghold secrets tune -max-lease-ttl=87600h ssh-host-signer
   ```

4. Create a role for signing host keys. Be sure to fill in the list of allowed domains, set `allow_bare_domains`, or both.

   ```
   $ d8 stronghold write ssh-host-signer/roles/hostrole \
       key_type=ca \
       algorithm_signer=rsa-sha2-256 \
       ttl=87600h \
       allow_host_certificates=true \
       allowed_domains="localdomain,example.com" \
       allow_subdomains=true
   ```

5. Sign the host's SSH public key.

```
$ d8 stronghold write ssh-host-signer/sign/hostrole \
    cert_type=host \
    public_key=@/etc/ssh/ssh_host_rsa_key.pub
Key             Value
---             -----
serial_number   3746eb17371540d9
signed_key      ssh-rsa-cert-v01@openssh.com AAAAHHNzaC1y...
```

6. Set the resulting signed certificate as `HostCertificate` in the SSH configuration on the host machine.

```
$ d8 stronghold write -field=signed_key ssh-host-signer/sign/hostrole \
    cert_type=host \
    public_key=@/etc/ssh/ssh_host_rsa_key.pub > /etc/ssh/ssh_host_rsa_key-
cert.pub
```

   Set permissions on the certificate to be `0640`:

```
chmod 0640 /etc/ssh/ssh_host_rsa_key-cert.pub
```

   Add host key and host certificate to the SSH configuration file.

```
# /etc/ssh/sshd_config
# ...

# For client keys
TrustedUserCAKeys /etc/ssh/trusted-user-ca-keys.pem

# For host keys
HostKey /etc/ssh/ssh_host_rsa_key
HostCertificate /etc/ssh/ssh_host_rsa_key-cert.pub
```

   Restart the SSH service to pick up the changes.

Client-Side host verification

1. Retrieve the host signing CA public key to validate the host signature of target machines.

```
curl http://127.0.0.1:8200/v1/ssh-host-signer/public_key
```

```
d8 stronghold read -field=public_key ssh-host-signer/config/ca
```

2. Add the resulting public key to the `known_hosts` file with authority.

```
# ~/.ssh/known_hosts
@cert-authority *.example.com ssh-rsa AAAAB3NzaC1yc2EAAA...
```

3. SSH into target machines as usual.

Troubleshooting

When initially configuring this type of key signing, enable `VERBOSE` SSH logging to help annotate any errors in the log.

```
# /etc/ssh/sshd_config
# ...
LogLevel VERBOSE
```

Restart SSH after making these changes.

By default, SSH logs to `/var/log/auth.log`, but so do many other things. To extract just the SSH logs, use the following:

```
tail -f /var/log/auth.log | grep --line-buffered "sshd"
```

If you are unable to make a connection to the host, the SSH server logs may provide guidance and insights.

Name is not a listed principal

If the `auth.log` displays the following messages:

```
# /var/log/auth.log
key_cert_check_authority: invalid certificate
Certificate invalid: name is not a listed principal
```

The certificate does not permit the username as a listed principal for authenticating to the system. This is most likely due to an OpenSSH bug (see Known issues for more information). This bug does not respect the `allowed_users` option value of “*”. Here are ways to work around this issue:

1. Set `default_user` in the role. If you are always authenticating as the same user, set the `default_user` in the role to the username you are SSHing into the target machine:

   ```
   $ d8 stronghold write ssh/roles/my-role -<<"EOH"
   {
     "default_user": "YOUR_USER",
     // ...
   }
   EOH
   ```

2. Set `valid_principals` during signing. In situations where multiple users may be authenticating to SSH vian Stronghold, set the list of valid principles during key signing to include the current username:

```
$ d8 stronghold write ssh-client-signer/sign/my-role -<<"EOH"
{
  "valid_principals": "my-user"
  // ...
}
EOH
```

No prompt after login

If you do not see a prompt after authenticating to the host machine, the signed certificate may not have the `permit-pty` extension. There are two ways to add this extension to the signed certificate.

- As part of the role creation

```
$ d8 stronghold write ssh-client-signer/roles/my-role -<<"EOH"
{
  "default_extensions": {
    "permit-pty": ""
  }
  // ...
}
EOH
```

- As part of the signing operation itself:

```
$ d8 stronghold write ssh-client-signer/sign/my-role -<<"EOH"
{
  "extensions": {
    "permit-pty": ""
  }
  // ...
}
EOH
```

No port forwarding

If port forwarding from the guest to the host is not working, the signed certificate may not have the `permit-port-forwarding` extension. Add the extension as part of the role creation or signing process to enable port forwarding. See No prompt after login for examples.

```
{
  "default_extensions": {
    "permit-port-forwarding": ""
  }
}
```

No x11 forwarding

If X11 forwarding from the guest to the host is not working, the signed certificate may not have the `permit-X11-forwarding` extension. Add the extension as part of the role creation or signing process to enable X11 forwarding. See No prompt after login for examples.

```
{
  "default_extensions": {
    "permit-X11-forwarding": ""
  }
}
```

No agent forwarding

If agent forwarding from the guest to the host is not working, the signed certificate may not have the `permit-agent-forwarding` extension. Add the extension as part of the role creation or signing process to enable agent forwarding. See No prompt after login for examples.

```
{
  "default_extensions": {
    "permit-agent-forwarding": ""
  }
}
```

Key comments

There are additional steps needed to preserve comment attributes in keys which ought to be considered if they are required. Private and public key may have comments applied to them and for example where `ssh-keygen` is used with its `-C` parameter - similar to:

```
ssh-keygen -C "...Comments" -N "" -t rsa -b 4096 -f host-ca
```

Adapted key values containing comments must be provided with the key related parameters as per the Stronghold CLI and API steps demonstrated below.

```
# Using CLI:
d8 stronghold secrets enable -path=hosts-ca ssh
KEY_PRI=$(cat ~/.ssh/id_rsa | sed -z 's/\n/\\n/g')
KEY_PUB=$(cat ~/.ssh/id_rsa.pub | sed -z 's/\n/\\n/g')
# Create / update keypair in Stronghold
d8 stronghold write ssh-client-signer/config/ca \
  generate_signing_key=false \
  private_key="${KEY_PRI}" \
  public_key="${KEY_PUB}"
```

```
# Using API:
curl -X POST -H "X-Vault-Token: ..." -d '{"type":"ssh"}' http://127.0.0.1:8200/v1/sys/
mounts/hosts-ca
KEY_PRI=$(cat ~/.ssh/id_rsa | sed -z 's/\n/\\n/g')
KEY_PUB=$(cat ~/.ssh/id_rsa.pub | sed -z 's/\n/\\n/g')
tee payload.json <<EOF
{
  "generate_signing_key" : false,
  "private_key"          : "${KEY_PRI}",
  "public_key"           : "${KEY_PUB}"
}
EOF
# Create / update keypair in Stronghold
curl -X POST -H "X-Vault-Token: ..." -d @payload.json http://127.0.0.1:8200/v1/hosts-
ca/config/ca
```

:::warning

IMPORTANT: Do NOT add a private key password since Stronghold can’t decrypt it. Destroy the keypair and `payload.json` from your hosts immediately after they have been confirmed as successfully uploaded.

:::

Known issues

- On SELinux-enforcing systems, you may need to adjust related types so that the SSH daemon is able to read it. For example, adjust the signed host certificate to be an `sshd_key_t` type.
- On some versions of SSH, you may get the following error:

  ```
  no separate private key for certificate
  ```

  This is a bug introduced in OpenSSH version 7.2 and fixed in 7.5. See OpenSSH bug 2617 for details.

- On some versions of SSH, you may get the following error on target host:

```
userauth_pubkey: certificate signature algorithm ssh-rsa: signature algorithm
not supported [preauth]
```

Fix is to add below line to /etc/ssh/sshd_config

```
CASignatureAlgorithms ^ssh-rsa
```

The ssh-rsa algorithm is no longer supported in OpenSSH 8.2

Transit Secrets engine

The transit secrets engine handles cryptographic functions on data in-transit. Stronghold doesn't store the data sent to the secrets engine. It can also be viewed as "cryptography as a service" or "encryption as a service". The transit secrets engine can also sign and verify data; generate hashes and HMACs of data; and act as a source of random bytes.

The primary use case for transit is to encrypt data from applications while still storing that encrypted data in some primary data store. This relieves the burden of proper encryption/decryption from application developers and pushes the burden onto Stronghold.

Key derivation is supported, which allows the same key to be used for multiple purposes by deriving a new key based on a user-supplied context value. In this mode, convergent encryption can optionally be supported, which allows the same input values to produce the same ciphertext.

Datakey generation allows processes to request a high-entropy key of a given bit length be returned to them, encrypted with the named key. Normally this will also return the key in plaintext to allow for immediate use, but this can be disabled to accommodate auditing requirements.

Working set management

The Transit engine supports versioning of keys. Key versions that are earlier than a key's specified `min_decryption_version` gets archived, and the rest of the key versions belong to the working set. This is a performance consideration to keep key loading fast, as well as a security consideration: by disallowing decryption of old versions of keys, found ciphertext corresponding to obsolete (but sensitive) data can not be decrypted by most users, but in an emergency the `min_decryption_version` can be moved back to allow for legitimate decryption.

Currently this archive is stored in a single storage entry. With some storage backends, notably those using Raft or Paxos for high-availability (HA) capabilities, frequent rotation may lead to a storage entry size for the archive that is larger than the storage backend can handle. For frequent rotation needs, using named keys that correspond to time bounds (for example, five-minute periods floored to the closest multiple of five) may provide a good alternative, allowing for several keys to be live at once and a deterministic way to decide which key to use at any given time.

NIST rotation guidance

Periodic rotation of the encryption keys is recommended, even in the absence of compromise. For AES-GCM keys, rotation should occur before approximately 2^{32} encryptions have been performed by a key version, following the guidelines of NIST publication 800-38D. It is recommended that operators estimate the encryption rate of a key and use that to determine a frequency of rotation that prevents the guidance limits from being reached. For example, if one determines that the estimated rate is 40 million operations per day, then rotating a key every three months is sufficient.

Key types

As of now, the transit secrets engine supports the following key types (all key types also generate separate HMAC keys):

- `aes128-gcm96` : AES-GCM with a 128-bit AES key and a 96-bit nonce; supports encryption, decryption, key derivation, and convergent encryption.
- `aes256-gcm96` : AES-GCM with a 256-bit AES key and a 96-bit nonce; supports encryption, decryption, key derivation, and convergent encryption (default).
- `chacha20-poly1305` : ChaCha20-Poly1305 with a 256-bit key; supports encryption, decryption, key derivation, and convergent encryption.
- `ed25519` : Ed25519; supports signing, signature verification, and key derivation.
- `ecdsa-p256` : ECDSA using curve P-256; supports signing and signature verification.
- `ecdsa-p384` : ECDSA using curve P-384; supports signing and signature verification.
- `ecdsa-p521` : ECDSA using curve P-521; supports signing and signature verification.
- `rsa-2048` : 2048-bit RSA key; supports encryption, decryption, signing, and signature verification.
- `rsa-3072` : 3072-bit RSA key; supports encryption, decryption, signing, and signature verification.
- `rsa-4096` : 4096-bit RSA key; supports encryption, decryption, signing, and signature verification.
- `hmac` : HMAC; supporting HMAC generation and verification.

> All key types support HMAC operations through the use of a second randomly generated key created key creation time or rotation. The HMAC key type only supports HMAC, and behaves identically to other algorithms with respect to the HMAC operations but supports key import. By default, the HMAC key type uses a 256-bit key.

RSA operations use one of the following methods:

- OAEP (encrypt, decrypt), with SHA-256 hash function and MGF
- PSS (sign, verify), with configurable hash function also used for MGF
- PKCS#1v1.5: (sign, verify), with configurable hash function

Convergent encryption

Convergent encryption is a mode where the same set of plain text and context always result in the same ciphertext. It does this by deriving a key using a key derivation function but also by deterministically

deriving a nonce. Because these properties differ for any combination of plaintext and ciphertext over a keyspace the size of 2^256, the risk of nonce reuse is near zero.

This has many practical uses. One common usage mode is to allow values to be stored encrypted in a database, but with limited lookup/query support, so that rows with the same value for a specific field can be returned from a query.

To accommodate for any needed upgrades to the algorithm, different versions of convergent encryption have historically been supported:

- Version 1 required the client to provide their own nonce, which is highly flexible but if done incorrectly can be dangerous. Keys using this version cannot be upgraded.
- Version 2 used an algorithmic approach to deriving the parameters. However, the algorithm used was susceptible to offline plaintext-confirmation attacks, which could allow attackers to brute force decryption if the plaintext size was small. Keys using version 2 can be upgraded by simply performing a rotate operation to a new key version; existing values can then be rewrapped against the new key version and will use the version 3 algorithm.
- Version 3 uses a different algorithm designed to be resistant to offline plaintext-confirmation attacks. It is similar to AES-SIV in that it uses a PRF to generate the nonce from the plaintext.

Setup

Most secrets engines must be configured in advance before they can perform their functions. These steps are usually completed by an operator or configuration management tool.

Enable the Transit secrets engine:

```
stronghold secrets enable transit
Success! Enabled the transit secrets engine at: transit/
```

By default, the secrets engine will mount at the name of the engine. To enable the secrets engine at a different path, use the `-path` argument.

Create a named encryption key:

```
stronghold write -f transit/keys/my-key
Success! Data written to: transit/keys/my-key
```

Usually each application has its own encryption key.

Usage

After the secrets engine is configured and a user/machine has a Stronghold token with the proper permission, it can use this secrets engine.

Encrypt some plaintext data using the `/encrypt` endpoint with a named key:

> All plaintext data **must be base64-encoded**. The reason for this requirement is that Stronghold does not require that the plaintext is “text”. It could be a binary file such as a PDF or image. The easiest safe transport mechanism for this data as part of a JSON payload is to base64-encode it.

```
stronghold write transit/encrypt/my-key plaintext=$(echo "my secret data" | base64)

Key           Value
---           -----
ciphertext    vault:v1:8SDd3WHDOjf7mq69CyCqYjBXAiQQAVZRkFM13ok481zoCmHnSeDX9vyf7w==
```

The returned ciphertext starts with `vault:v1:`. The first prefix (`stronghold`) identifies that it has been wrapped by Stronghold. The `v1` indicates the key version 1 was used to encrypt the plaintext; therefore, when you rotate keys, Stronghold knows which version to use for decryption.

Note that Stronghold *does not store* any of this data. The caller is responsible for storing the encrypted ciphertext. When the caller wants the plaintext, it must provide the ciphertext back to Stronghold to decrypt the value.

> Stronghold HTTP API imposes a maximum request size of 32MB to prevent a DoS attack. This can be adjusted in the `listener` block in the Stronghold configuration.

Decrypt a piece of data using the `/decrypt` endpoint with a named key:

```
stronghold write transit/decrypt/my-key ciphertext=vault:v1:8SDd3WHDOjf7mq69CyCqYjBXAi
QQAVZRkFM13ok481zoCmHnSeDX9vyf7w==

Key          Value
---          -----
plaintext    bXkgc2VjcmV0IGRhdGEK
```

The resulting data is base64-encoded. Decode it to get the raw plaintext:

```
base64 --decode <<< "bXkgc2VjcmV0IGRhdGEK"
my secret data
```

Or in one command:

```
stronghold write -field=plaintext transit/decrypt/my-key ciphertext=... | base64 --
decode
my secret data
```

Using ACLs, it is possible to restrict using the transit secrets engine such that trusted operators can manage the named keys, and applications can only encrypt or decrypt using the named keys they need access to.

Rotate the underlying encryption key. This will generate a new encryption key and add it to the keyring for the named key:

```
stronghold write -f transit/keys/my-key/rotate
Success! Data written to: transit/keys/my-key/rotate
```

Future encryptions will use this new key. Old data can still be decrypted due to the use of a key ring.

Upgrade already-encrypted data to a new key. Stronghold will decrypt the value using the appropriate key in the keyring and then encrypt the resulting plaintext with the newest key in the keyring.

```
stronghold write transit/rewrap/my-key ciphertext=vault:v1:8SDd3WHDOjf7mq69CyCqYjBXAiQ
QAVZRkFM13ok481zoCmHnSeDX9vyf7w==

Key           Value
---           -----
ciphertext    vault:v2:...
```

This process **does not reveal** the plaintext data. As such, a Stronghold policy could grant almost an untrusted process the ability to “rewrap” encrypted data, since the process would not be able to get access to the plaintext data.

Importing custom key (Bring Your Own Key, BYOK)

> ⚠ Key import functionality supports cases in which there is a need to bring in an existing key from an HSM or other outside system. It is more secure to have Transit generate and manage a key within Stronghold.

First, the wrapping key needs to be read from transit:

```
stronghold read transit/wrapping_key
```

The wrapping key will be a 4096-bit RSA public key.

Then the wrapping key is used to create the ciphertext input for the `import` endpoint, as described below. In the below, the target key refers to the key being imported.

HSM

If the key is being imported from an HSM that supports PKCS#11, there are two possible scenarios:

- If the HSM supports the CKM_RSA_AES_KEY_WRAP mechanism, that can be used to wrap the target key using the wrapping key.
- Otherwise, two mechanisms can be combined to wrap the target key. First, a 256-bit AES key should be generated and then used to wrap the target key using the CKM_AES_KEY_WRAP_KWP mechanism. Then the AES key should be wrapped under the wrapping key using the CKM_RSA_PKCS_OAEP mechanism using MGF1 and either SHA-1, SHA-224, SHA-256, SHA-384, or SHA-512.

The ciphertext is constructed by appending the wrapped target key to the wrapped AES key.

The ciphertext bytes should be base64-encoded.

Manual process

If the target key is not stored in an HSM or KMS, the following steps can be used to construct the ciphertext for the input of the `import` endpoint:

- Generate an ephemeral 256-bit AES key.
- Wrap the target key using the ephemeral AES key with AES-KWP.

> When wrapping a symmetric key (such as an AES or ChaCha20 key), wrap the raw bytes of the key. For instance, with an AES 128-bit key, this'll be a byte array 16 characters in length that will directly be wrapped without base64 or other encodings.
>
> When wrapping an asymmetric key (such as a RSA or ECDSA key), wrap the PKCS8 encoded format of this key, in raw DER/binary form. Do not apply PEM encoding to this blob prior to encryption and do not base64 encode it.

- Wrap the AES key under the Stronghold wrapping key using RSAES-OAEP with MGF1 and either SHA-1, SHA-224, SHA-256, SHA-384, or SHA-512.
- Delete the ephemeral AES key.
- Append the wrapped target key to the wrapped AES key.
- Base64 encode the result.

TOTP Secrets engine

The TOTP secrets engine generates time-based credentials according to the TOTP standard. The secrets engine can also be used to generate a new key and validate passwords generated by that key.

The TOTP secrets engine can act as both a generator (like Google Authenticator) and a provider (like the Google.com sign in service).

As a generator

The TOTP secrets engine can act as a TOTP code generator. In this mode, it can replace traditional TOTP generators like Google Authenticator. It provides an added layer of security since the ability to generate codes is guarded by policies and the entire process is audited.

Setup

Most secrets engines must be configured in advance before they can perform their functions. These steps are usually completed by an operator or configuration management tool.

1. Enable the TOTP secrets engine:

   ```
   $ d8 stronghold secrets enable totp
   Success! Enabled the totp secrets engine at: totp/
   ```

   By default, the secrets engine will mount at the name of the engine. To enable the secrets engine at a different path, use the `-path` argument.

2. Configure a named key. The name of this key will be a human identifier as to its purpose.

   ```
   $ d8 stronghold write totp/keys/my-key \
       url="otpauth://totp/Stronghold:test@test.com?
   secret=Y64VEVMBTSXCYIWRSHRNDZW62MPGVU2G&issuer=Stronghold"
   Success! Data written to: totp/keys/my-key
   ```

   The `url` corresponds to the secret key or value from the barcode provided by the third-party service.

Usage

After the secrets engine is configured and a user/machine has an Stronghold token with the proper permission, it can generate credentials.

1. Generate a new time-based OTP by reading from the `/code` endpoint with the name of the key:

   ```
   $ d8 stronghold read totp/code/my-key
   Key     Value
   ---     -----
   code    260610
   ```

   Using ACLs, it is possible to restrict using the TOTP secrets engine such that trusted operators can manage the key definitions, and both users and applications are restricted in the credentials they are allowed to read.

As a provider

The TOTP secrets engine can also act as a TOTP provider. In this mode, it can be used to generate new keys and validate passwords generated using those keys.

Setup

Most secrets engines must be configured in advance before they can perform their functions. These steps are usually completed by an operator or configuration management tool.

1. Enable the TOTP secrets engine:

   ```
   $ d8 stronghold secrets enable totp
   Success! Enabled the totp secrets engine at: totp/
   ```

   By default, the secrets engine will mount at the name of the engine. To enable the secrets engine at a different path, use the `-path` argument.

2. Create a named key, using the `generate` option. This tells Stronghold to be the provider:

   ```
   $ d8 stronghold write totp/keys/my-user \
       generate=true \
       issuer=Stronghold \
       account_name=user@test.com

   Key         Value
   ---         -----
   barcode     iVBORw0KGgoAAAANSUhEUgAAAMgAAADIEAAAAADYoy0BA...
   url         otpauth://totp/Stronghold:user@test.com?
   algorithm=SHA1&digits=6&issuer=Stronghold&period=30&secret=V7MBSK324I7KF6KVW34ND
   FH2GYHIF6JY
   ```

   The response includes a base64-encoded barcode and OTP url. Both are equivalent. Give these to the user who is authenticating with TOTP.

Usage

1. As a user, validate a TOTP code generated by a third-party app:

   ```
   $ d8 stronghold write totp/code/my-user code=886531
   Key      Value
   ---      -----
   valid    true
   ```

Kubernetes Secrets engine

The Kubernetes Secrets Engine for Stronghold generates Kubernetes service account tokens (not to be confused with Stronghold tokens), and optionally service accounts, role bindings, and roles. The created service account tokens have a configurable TTL and any objects created are automatically deleted when the Stronghold lease expires.

For each lease, Stronghold will create a service account token attached to the defined service account. The service account token is returned to the caller.

To learn more about service accounts in Kubernetes, visit the Kubernetes service account and Kubernetes RBAC documentation.

> ⚠ **Note:** We do not recommend using tokens created by the Kubernetes Secrets Engine to authenticate with the Stronghold Kubernetes Auth Method. This will generate many unique identities in Stronghold that will be hard to manage.

Setup

The Kubernetes Secrets Engine must be configured in advance before it can perform its functions. These steps are usually completed by an operator or configuration management tool.

1. By default, Stronghold will connect to Kubernetes using its own service account. If using the standard Helm chart, this service account is created automatically by default and named after the Helm release (often `stronghold`, but this can be configured via the Helm value `server.serviceAccount.name`).

   It's necessary to ensure that the service account Stronghold uses will have permissions to manage service account tokens, and optionally manage service accounts, roles, and role bindings. These permissions can be managed using a Kubernetes role or cluster role. The role is attached to the Stronghold service account with a role binding or cluster role binding.

   For example, a minimal cluster role to create service account tokens is:

   ```
   apiVersion: rbac.authorization.k8s.io/v1
   kind: ClusterRole
   metadata:
     name: k8s-minimal-secrets-abilities
   rules:
   - apiGroups: [""]
     resources: ["serviceaccounts/token"]
     verbs: ["create"]
   ```

   Similarly, you can create a more permissive cluster role with full permissions to manage tokens, service accounts, bindings, and roles.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: k8s-full-secrets-abilities
rules:
- apiGroups: [""]
  resources: ["serviceaccounts", "serviceaccounts/token"]
  verbs: ["create", "update", "delete"]
- apiGroups: ["rbac.authorization.k8s.io"]
  resources: ["rolebindings", "clusterrolebindings"]
  verbs: ["create", "update", "delete"]
- apiGroups: ["rbac.authorization.k8s.io"]
  resources: ["roles", "clusterroles"]
  verbs: ["bind", "escalate", "create", "update", "delete"]
```

Create this role in Kubernetes (e.g., with `d8 k apply -f`).

Moreover, if you want to use label selection to configure the namespaces on which a role can act, you will need to grant Stronghold permissions to read namespaces.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: k8s-full-secrets-abilities-with-labels
rules:
- apiGroups: [""]
  resources: ["namespaces"]
  verbs: ["get"]
- apiGroups: [""]
  resources: ["serviceaccounts", "serviceaccounts/token"]
  verbs: ["create", "update", "delete"]
- apiGroups: ["rbac.authorization.k8s.io"]
  resources: ["rolebindings", "clusterrolebindings"]
  verbs: ["create", "update", "delete"]
- apiGroups: ["rbac.authorization.k8s.io"]
  resources: ["roles", "clusterroles"]
  verbs: ["bind", "escalate", "create", "update", "delete"]
```

Note: Getting the right permissions for Stronghold will require some trial and error most likely since Kubernetes has strict protections against privilege escalation. You can read more in the Kubernetes RBAC documentation.

Note: Protect the Stronghold service account, especially if you use broader permissions for it, as it is essentially a cluster administrator account.

1. Create a role binding to bind the role to Stronghold's service account and grant Stronghold permission to manage tokens.

   ```
   apiVersion: rbac.authorization.k8s.io/v1
   kind: ClusterRoleBinding
   metadata:
     name: stronghold-token-creator-binding
   roleRef:
     apiGroup: rbac.authorization.k8s.io
     kind: ClusterRole
     name: k8s-minimal-secrets-abilities
   subjects:
   - kind: ServiceAccount
     name: stronghold
     namespace: stronghold
   ```

   For more information on Kubernetes roles, service accounts, bindings, and tokens, visit the Kubernetes RBAC documentation.

2. If Stronghold will not be automatically managing roles or service accounts (see Automatically Managing Roles and Service Accounts), then you will need to set up a service account that Stronghold will issue tokens for.

> ⚠ **Note**: It is highly recommended that the service account that Stronghold issues tokens for is **NOT** the same service account that Stronghold itself uses.

The examples we will use will under the namespace `test`, which you can create if it does not already exist.

```
$ d8 k create namespace test
namespace/test created
```

Here is a simple set up of a service account, role, and role binding in the Kubernetes `test` namespace with basic permissions we will use for this document:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: test-service-account-with-generated-token
  namespace: test
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: test-role-list-pods
  namespace: test
rules:
- apiGroups: [""]
  resources: ["pods"]
  verbs: ["list"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: test-role-abilities
  namespace: test
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: test-role-list-pods
subjects:
- kind: ServiceAccount
  name: test-service-account-with-generated-token
  namespace: test
```

You can create these objects with `d8 k apply -f`.

1. Enable the Kubernetes Secrets Engine:

   ```
   $ d8 stronghold secrets enable kubernetes
   Success! Enabled the kubernetes Secrets Engine at: kubernetes/
   ```

   By default, the secrets engine will mount at the same name as the engine, i.e., `kubernetes/` here. This can be changed by passing the `-path` argument when enabling.

2. Configure the mount point. An empty config is allowed.

   ```
   d8 stronghold write -f kubernetes/config
   ```

3. You can now configure Kubernetes Secrets Engine to create an Stronghold role (**not** the same as a Kubernetes role) that can generate service account tokens for the given service account:

```
$ d8 stronghold write kubernetes/roles/my-role \
    allowed_kubernetes_namespaces="*" \
    service_account_name="test-service-account-with-generated-token" \
    token_default_ttl="10m"
```

Generating credentials

After a user has authenticated to Stronghold and has sufficient permissions, a write to the `creds` endpoint for the Stronghold role will generate and return a new service account token.

```
$ d8 stronghold write kubernetes/creds/my-role \
    kubernetes_namespace=test

Key                        Value
---                        -----
lease_id                   kubernetes/creds/my-role/31d771a6-...
lease_duration             10m0s
lease_renwable             false
service_account_name       test-service-account-with-generated-token
service_account_namespace  test
service_account_token      eyJHbGci0iJSUzI1NiIsImtpZCI6ImlrUEE...
```

You can use the service account token above (`eyJHbG...`) with any Kubernetes API request that its service account is authorized for (through role bindings).

```
$ curl -sk $(d8 k config view --minify -o 'jsonpath={.clusters[].cluster.server}')/
api/v1/namespaces/test/pods \
    --header "Authorization: Bearer eyJHbGci0iJSUzI1Ni..."
{
  "kind": "PodList",
  "apiVersion": "v1",
  "metadata": {
    "resourceVersion": "1624"
  },
  "items": []
}
```

When the lease expires, you can verify that the token has been revoked.

```
$ curl -sk $(d8 k config view --minify -o 'jsonpath={.clusters[].cluster.server}')/
api/v1/namespaces/test/pods \
    --header "Authorization: Bearer eyJHbGci0iJSUzI1Ni..."
{
  "kind": "Status",
  "apiVersion": "v1",
  "metadata": {},
  "status": "Failure",
  "message": "Unauthorized",
  "reason": "Unauthorized",
  "code": 401
}
```

TTL

Kubernetes service account tokens have a time-to-live (TTL). When a token expires it is automatically revoked.

You can set a default (`token_default_ttl`) and a maximum TTL (`token_max_ttl`) when creating or tuning the Stronghold role.

```
$ d8 stronghold write kubernetes/roles/my-role \
    allowed_kubernetes_namespaces="*" \
    service_account_name="new-service-account-with-generated-token" \
    token_default_ttl="10m" \
    token_max_ttl="2h"
```

You can also set a TTL (`ttl`) when you generate the token from the credentials endpoint. The TTL of the token will be given the default if not specified (and cannot exceed the maximum TTL of the role, if present).

```
$ d8 stronghold write kubernetes/creds/my-role \
    kubernetes_namespace=test \
    ttl=20m

Key                        Value
---                        -----
lease_id                   kubernetes/creds/my-role/31d771a6-...
lease_duration             20m0s
lease_renwable             false
service_account_name       new-service-account-with-generated-token
service_account_namespace  test
service_account_token      eyJHbGci0iJSUzI1NiIsImtpZCI6ImlrUEE...
```

You can verify the token’s TTL by decoding the JWT token and extracting the `iat` (issued at) and `exp` (expiration time) claims.

```
$ echo 'eyJhbGc...' | cut -d'.' -f2 | base64 -d  | jq -r '.iat,.exp|todate'
2022-05-20T17:14:50Z
2022-05-20T17:34:50Z
```

Audiences

Kubernetes service account tokens have audiences.

You can set default audiences (`token_default_audiences`) when creating or tuning the Stronghold role. The Kubernetes cluster default audiences for service account tokens will be used if not specified.

```
$ d8 stronghold write kubernetes/roles/my-role \
    allowed_kubernetes_namespaces="*" \
    service_account_name="new-service-account-with-generated-token" \
    token_default_audiences="custom-audience"
```

You can also set audiences (`audiences`) when you generate the token from the credentials endpoint. The audiences of the token will be given the default audiences if not specified.

```
$ d8 stronghold write kubernetes/creds/my-role \
    kubernetes_namespace=test \
    audiences="another-custom-audience"

Key                         Value
---                         -----
lease_id                    kubernetes/creds/my-role/SriWQf0bPZ...
lease_duration              768h
lease_renwable              false
service_account_name        new-service-account-with-generated-token
service_account_namespace   test
service_account_token       eyJHbGci0iJSUzI1NiIsImtpZCI6ImlrUEE...
```

You can verify the token’s audiences by decoding the JWT.

```
$ echo 'eyJhbGc...' | cut -d'.' -f2 | base64 -d
{"aud":["another-custom-audience"]...
```

Automatically managing roles and service accounts

When configuring the Stronghold role, you can pass in parameters to specify that you want to automatically generate the Kubernetes service account and role binding, and optionally generate the Kubernetes role itself.

If you want to configure the Stronghold role to use a pre-existing Kubernetes role, but generate the service account and role binding automatically, you can set the `kubernetes_role_name` parameter.

```
$ d8 stronghold write kubernetes/roles/auto-managed-sa-role \
    allowed_kubernetes_namespaces="test" \
    kubernetes_role_name="test-role-list-pods"
```

> ⚠ **Note**: Stronghold's service account will also need access to the resources it is granting access to. This can be done for the examples above with `d8 k -n test create rolebinding --role test-role-list-pods --serviceaccount=d8-stronghold:stronghold stronghold-test-role-abilities`. This is how Kubernetes prevents privilege escalation. You can read more in the Kubernetes RBAC documentation.

You can then get credentials with the automatically generated service account.

```
$ d8 stronghold write kubernetes/creds/auto-managed-sa-role \
    kubernetes_namespace=test
Key                         Value
---                         -----
lease_id                    kubernetes/creds/auto-managed-sa-role/
cujRLYjKZUMQk6dkHBGGWm67
lease_duration              768h
lease_renewable             false
service_account_name        v-token-auto-man-1653001548-5z6hrgsxnmzncxejztml4arz
service_account_namespace   test
service_account_token       eyJHbGci0iJSUzI1Ni...
```

Furthermore, Stronghold can also automatically create the role in addition to the service account and role binding by specifying the `generated_role_rules` parameter, which accepts a set of JSON or YAML rules for the generated role.

```
$ d8 stronghold write kubernetes/roles/auto-managed-sa-and-role \
    allowed_kubernetes_namespaces="test" \
    generated_role_rules='{"rules":[{"apiGroups":[""],"resources":["pods"],"verbs":
["list"]}]}'
```

You can then get credentials in the same way as before.

```
$ d8 stronghold write kubernetes/creds/auto-managed-sa-and-role \
    kubernetes_namespace=test
Key                          Value
---                          -----
lease_id                     kubernetes/creds/auto-managed-sa-and-role/
pehLtegoTP8vCkcaQozUqOHf
lease_duration               768h
lease_renewable              false
service_account_name         v-token-auto-man-1653002096-4imxf3ytjh5hbyro9s1oqdo3
service_account_namespace    test
service_account_token        eyJHbGci0iJSUzI1Ni...
```

LDAP Secrets engine

The LDAP secrets engine provides management of LDAP credentials as well as dynamic creation of credentials. It supports integration with implementations of the LDAP v3 protocol, including OpenLDAP, ALD Pro, Active Directory, and IBM Resource Access Control Facility (RACF).

The secrets engine has three primary features:

- Static Credentials
- Dynamic Credentials
- Service Account Check-Out

Setup

1. Enable the LDAP secret engine:

   ```
   d8 stronghold secrets enable ldap
   ```

   By default, the secrets engine will mount at the name of the engine. To enable the secrets engine at a different path, use the `-path` argument.

2. Configure the credentials that Stronghold uses to communicate with LDAP to generate passwords:

   ```
   $ d8 stronghold write ldap/config \
       binddn=$USERNAME \
       bindpass=$PASSWORD \
       url=ldaps://138.91.247.105
   ```

   Note: it's recommended a dedicated entry management account be created specifically for Stronghold.

3. Rotate the root password so only Stronghold knows the credentials:

   ```
   d8 stronghold write -f ldap/rotate-root
   ```

> Note: it’s not possible to retrieve the generated password once rotated by Stronghold. It’s recommended a dedicated entry management account be created specifically for Stronghold.

Schemas

The LDAP Secret Engine supports three different schemas:

- `openldap` (default)
- `racf`
- `ad`

OpenLDAP

By default, the LDAP Secret Engine assumes the entry password is stored in `userPassword`. There are many object classes that provide `userPassword` including for example:

- `organization`
- `organizationalUnit`
- `organizationalRole`
- `inetOrgPerson`
- `person`
- `posixAccount`

Resource access control facility (RACF)

For managing IBM’s Resource Access Control Facility (RACF) security system, the secret engine must be configured to use the schema `racf`.

Generated passwords must be 8 characters or less to support RACF. The length of the password can be configured using a password policy:

```
$ d8 stronghold write ldap/config \
 binddn=$USERNAME \
 bindpass=$PASSWORD \
 url=ldaps://138.91.247.105 \
 schema=racf \
 password_policy=racf_password_policy
```

Active directory (AD)

For managing Active Directory instances, the secret engine must be configured to use the schema `ad`.

```
$ d8 stronghold write ldap/config \
 binddn=$USERNAME \
 bindpass=$PASSWORD \
 url=ldaps://138.91.247.105 \
 schema=ad
```

Static credentials

Setup

1. Configure a static role that maps a name in Stronghold to an entry in LDAP. Password rotation settings will be managed by this role.

```
$ d8 stronghold write ldap/static-role/lf-edge\
    dn='uid=lf-edge,ou=users,dc=lf-edge,dc=com' \
    username='stronghold'\
    rotation_period="24h"
```

2. Request credentials for the “stronghold” role:

```
d8 stronghold read ldap/static-cred/lf-edge
```

Password rotation

Passwords can be managed in two ways:

- automatic time based rotation
- manual rotation

Auto password rotation

Passwords will automatically be rotated based on the `rotation_period` configured in the static role (minimum of 5 seconds). When requesting credentials for a static role, the response will include the time before the next rotation (`ttl`).

Auto-rotation is currently only supported for static roles. The `binddn` account used by Stronghold should be rotated using the `rotate-root` endpoint to generate a password only Stronghold will know.

Manual rotation

Static roles can be manually rotated using the `rotate-role` endpoint. When manually rotated the rotation period will start over.

Deleting static roles

Passwords are not rotated upon deletion of a static role. The password should be manually rotated prior to deleting the role or revoking access to the static role.

Dynamic credentials

Setup

Dynamic credentials can be configured by calling the `/role/:role_name` endpoint:

```
$ d8 stronghold write ldap/role/dynamic-role \
  creation_ldif=@/path/to/creation.ldif \
  deletion_ldif=@/path/to/deletion.ldif \
  rollback_ldif=@/path/to/rollback.ldif \
  default_ttl=1h \
  max_ttl=24h
```

> The `rollback_ldif` argument is optional, but recommended. The statements within `rollback_ldif` will be executed if the creation fails for any reason. This ensures any entities are removed in the event of a failure.

To generate credentials:

```
$ d8 stronghold read ldap/creds/dynamic-role
Key                    Value
---                    -----
lease_id               ldap/creds/dynamic-role/HFgd6uKaDomVMvJpYbn9q4q5
lease_duration         1h
lease_renewable        true
distinguished_names    [cn=v_token_dynamic-role_FfH2i1c4dO_1611952635,ou=users,dc=lear
n,dc=example]
password
xWMjkIFMerYttEbzfnBVZvhRQGmhpAA0yeTya8fdmDB3LXDzGrjNEPV2bCPE9CW6
username               v_token_testrole_FfH2i1c4dO_1611952635
```

The `distinguished_names` field is an array of DNs that are created from the `creation_ldif` statements. If more than one LDIF entry is included, the DN from each statement will be included in this field. Each entry in this field corresponds to a single LDIF statement. No de-duplication occurs and order is maintained.

LDIF entries

User account management is provided through LDIF entries. The LDIF entries may be a base64-encoded version of the LDIF string. The string will be parsed and validated to ensure that it adheres to LDIF syntax. A good reference for proper LDIF syntax can be found in the LDIF format reference.

Some important things to remember when crafting your LDIF entries:

- There should not be any trailing spaces on any line, including empty lines
- Each `modify` block needs to be preceded with an empty line
- Multiple modifications for a `dn` can be defined in a single `modify` block. Each modification needs to close with a single dash (`-`)

Active directory (AD)

For Active Directory, there are a few additional details that are important to remember:

To create a user programmatically in AD, you first `add` a user object and then `modify` that user to provide a password and enable the account.

- Passwords in AD are set using the `unicodePwd` field. This must be proceeded by two (2) colons (`::`).
- When setting a password programmatically in AD, the following criteria must be met:
  - The password must be enclosed in double quotes (`" "`)
  - The password must be in `UTF16LE` format
  - The password must be `base64` -encoded
  - Additional details can be found in the Microsoft guide to set user password with LDIFDE
- Once a user's password has been set, it can be enabled. AD uses the `userAccountControl` field for this purpose:
  - To enable the account, set `userAccountControl` to `512`
  - You will likely also want to disable AD's password expiration for this dynamic user account. The `userAccountControl` value for this is: `65536`
  - `userAccountControl` flags are cumulative, so to set both of the above two flags, add up the two values (`512 + 65536 = 66048`): set `userAccountControl` to `66048`
  - See userAccountControl property flag descriptions for details on `userAccountControl` flags

`sAMAccountName` is a common field when working with AD users. It is used to provide compatibility with legacy Windows NT systems and has a limit of 20 characters. Keep this in mind when defining your `username_template` . See sAMAccountName attribute documentation for additional details.

Since the default `username_template` is longer than 20 characters which follows the template of `v_{{.DisplayName}}_{{.RoleName}}_{{random 10}}_{{unix_time}}` , we recommend customising the `username_template` on the role configuration to generate accounts with names less than 20 characters.

With regard to adding dynamic users to groups, AD doesn't let you directly modify a user's `memberOf` attribute. The `member` attribute of a group and `memberOf` attribute of a user are linked attributes. Linked attributes are forward link/back link pairs, with the forward link able to be modified. In the case of AD group membership, the group's `member` attribute is the forward link. In order to add a newly-created dynamic user to a group, we also need to issue a `modify` request to the desired group and update the group membership with the new user.

Active directory LDIF example

The various `*_ldif` parameters are templates that use the go template language. A complete LDIF example for creating an Active Directory user account is provided here for reference:

```
dn: CN={{.Username}},OU=Stronghold,DC=adtesting,DC=lab
changetype: add
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: user
userPrincipalName: {{.Username}}@adtesting.lab
sAMAccountName: {{.Username}}

dn: CN={{.Username}},OU=Stronghold,DC=adtesting,DC=lab
changetype: modify
replace: unicodePwd
unicodePwd::{{ printf "%q" .Password | utf16le | base64 }}
-
replace: userAccountControl
userAccountControl: 66048
-

dn: CN=test-group,OU=Stronghold,DC=adtesting,DC=lab
changetype: modify
add: member
member: CN={{.Username}},OU=Stronghold,DC=adtesting,DC=lab
-
```

Service account Check-Out

Service account check-out provides a library of service accounts that can be checked out by a person or by machines. Stronghold will automatically rotate the password each time a service account is checked in. Service accounts can be voluntarily checked in, or Stronghold will check them in when their lending period (or, “ttl”, in Stronghold’s language) ends.

The service account check-out functionality works with various schemas, including OpenLDAP, Active Directory, and RACF. In the following usage example, the secrets engine is configured to manage a library of service accounts in an Active Directory instance.

First we’ll need to enable the LDAP secrets engine and tell it how to securely connect to an AD server.

```
$ d8 stronghold secrets enable ldap
Success! Enabled the ad secrets engine at: ldap/

$ d8 stronghold write ldap/config \
    binddn=$USERNAME \
    bindpass=$PASSWORD \
    url=ldaps://138.91.247.105 \
    userdn='dc=example,dc=com'
```

Our next step is to designate a set of service accounts for check-out.

```
$ d8 stronghold write ldap/library/accounting-team \
    service_account_names=fizz@example.com,buzz@example.com \
    ttl=10h \
    max_ttl=20h \
    disable_check_in_enforcement=false
```

In this example, the service account names of `fizz@example.com` and `buzz@example.com` have already been created on the remote AD server. They've been set aside solely for Stronghold to handle. The `ttl` is how long each check-out will last before Stronghold checks in a service account, rotating its password during check-in. The `max_ttl` is the maximum amount of time it can live if it's renewed. Both parameters are set to `24h` by default. Also by default, a service account must be checked in by the same Stronghold entity or client token that checked it out. However, if this behavior causes problems, set `disable_check_in_enforcement=true`.

When a library of service accounts has been created, view their status at any time to see if they're available or checked out.

```
$ d8 stronghold read ldap/library/accounting-team/status
Key                 Value
---                 -----
buzz@example.com    map[available:true]
fizz@example.com    map[available:true]
```

To check out any service account that's available, simply execute:

```
$ d8 stronghold write -f ldap/library/accounting-team/check-out
Key                     Value
---                     -----
lease_id                ldap/library/accounting-team/check-out/
EpuS8cX7uEsDzOwW9kkKOyGW
lease_duration          10h
lease_renewable         true
password                ?@09AZKh03hBORZPJcTDgLfntlHqxLy29tcQjPVThzuwWAx/
Twx4a2ZcRQRqrZ1w
service_account_name    fizz@example.com
```

If the default `ttl` for the check-out is higher than needed, set the check-out to last for a shorter time by using:

```
$ d8 stronghold write ldap/library/accounting-team/check-out ttl=30m
Key                    Value
---                    -----
lease_id               ldap/library/accounting-team/check-out/
gMonJ2jB6kYs6d3Vw37WFDCY
lease_duration         30m
lease_renewable        true
password               ?@09AZerLLuJfEMbRqP+3yfQYDSq6laP48TCJRBJaJu/kDKLsq9WxL9szVAvL/
E1
service_account_name   buzz@example.com
```

This can be a nice way to say, “Although I *can* have a check-out for 24 hours, if I haven’t checked it in after 30 minutes, I forgot or I’m a dead instance, so you can just check it back in.”

If no service accounts are available for check-out, Stronghold will return a 400 Bad Request.

```
$ d8 stronghold write -f ldap/library/accounting-team/check-out
Error writing data to ldap/library/accounting-team/check-out: Error making API
request.

URL: POST http://localhost:8200/v1/ldap/library/accounting-team/check-out
Code: 400. Errors:

* No service accounts available for check-out.
```

To extend a check-out, renew its lease.

```
$ d8 stronghold lease renew ldap/library/accounting-team/check-out/
0C2wmeaDmsToVFc0zDiX9cMq
Key                Value
---                -----
lease_id           ldap/library/accounting-team/check-out/0C2wmeaDmsToVFc0zDiX9cMq
lease_duration     10h
lease_renewable    true
```

Renewing a check-out means its current password will live longer, since passwords are rotated anytime a password is *checked in* either by a caller, or by Stronghold because the check-out `ttl` ends.

To check a service account back in for others to use, call:

```
$ d8 stronghold write -f ldap/library/accounting-team/check-in
Key          Value
---          -----
check_ins    [fizz@example.com]
```

Most of the time this will just work, but if multiple service accounts are checked out by the same caller, Stronghold will need to know which one(s) to check in.

```
$ d8 stronghold write ldap/library/accounting-team/check-in service_account_names=fizz
@example.com
Key          Value
---          -----
check_ins    [fizz@example.com]
```

To perform a check-in, Stronghold verifies that the caller *should* be able to check in a given service account. To do this, Stronghold looks for either the same entity ID used to check out the service account, or the same client token.

If a caller is unable to check in a service account, or simply doesn't try, Stronghold will check it back in automatically when the `ttl` expires. However, if that is too long, service accounts can be forcibly checked in by a highly privileged user through:

```
$ d8 stronghold write -f ldap/library/manage/accounting-team/check-in
Key          Value
---          -----
check_ins    [fizz@example.com]
```

Or, alternatively, revoking the secret's lease has the same effect.

```
$ d8 stronghold lease revoke ldap/library/accounting-team/check-out/
PvBVG0m7pEg2940Cb3Jw3KpJ
All revocation operations queued successfully!
```

LDAP password policy

The LDAP secret engine does not hash or encrypt passwords prior to modifying values in LDAP. This behavior can cause plaintext passwords to be stored in LDAP.

To avoid having plaintext passwords stored, the LDAP server should be configured with an LDAP password policy (ppolicy, not to be confused with an Stronghold password policy). A ppolicy can enforce rules such as hashing plaintext passwords by default.

The following is an example of an LDAP password policy to enforce hashing on the data information tree (DIT) `dc=example,dc=com`:

```
dn: cn=module{0},cn=config
changetype: modify
add: olcModuleLoad
olcModuleLoad: ppolicy

dn: olcOverlay={2}ppolicy,olcDatabase={1}mdb,cn=config
changetype: add
objectClass: olcPPolicyConfig
objectClass: olcOverlayConfig
olcOverlay: {2}ppolicy
olcPPolicyDefault: cn=default,ou=pwpolicies,dc=example,dc=com
olcPPolicyForwardUpdates: FALSE
olcPPolicyHashCleartext: TRUE
olcPPolicyUseLockout: TRUE
```

Identity Secrets engine

Overview

The Identity secrets engine is the identity management solution for Stronghold. It internally maintains the clients who are recognized by Stronghold. Each client is internally termed as an `Entity`. An entity can have multiple `Aliases`. For example, a single user who has accounts in both GitLab and LDAP, can be mapped to a single entity in Stronghold that has 2 aliases, one of type GitLab and one of type LDAP. When a client authenticates via any of the credential backends (except the Token backend), Stronghold creates a new entity and attaches a new alias to it, if a corresponding entity doesn't already exist. The entity identifier will be tied to the authenticated token. When such tokens are put to use, their entity identifiers are audit logged, marking a trail of actions performed by specific users.

Identity store allows operators to **manage** the entities in Stronghold. Entities can be created and aliases can be tied to entities, via the ACL'd API. There can be policies set on the entities which adds capabilities to the tokens that are tied to entity identifiers. The capabilities granted to tokens via the entities are **an addition** to the existing capabilities of the token and **not** a replacement. The capabilities of the token that get inherited from entities are computed dynamically at request time. This provides flexibility in controlling the access of tokens that are already issued.

> ⚠ **NOTE:** This secrets engine will be mounted by default. This secrets engine cannot be disabled or moved. For more conceptual overview on identity, refer to the Identity documentation.

The Stronghold Identity secrets engine supports several different features. Each one is individually documented on its own page.

- Identity tokens
- OIDC Identity Provider

OIDC identity provider

Stronghold is an OpenID Connect (OIDC) identity provider. This enables client applications that speak the OIDC protocol to leverage Stronghold's source of identity and wide range of authentication methods when authenticating end-users. Client applications can configure their authentication logic to talk to Stronghold. Once enabled, Stronghold will act as the bridge to other identity providers via its existing authentication methods. Client applications can also obtain identity information for their end-users by leveraging custom templating of Stronghold identity information.

Setup

The Stronghold OIDC provider system is built on top of the identity secrets engine. This secrets engine is mounted by default and cannot be disabled or moved.

Each Stronghold namespace has a default OIDC provider and a key. This built-in configuration enables client applications to begin using Stronghold as a source of identity with minimal configuration.

The following steps show a minimal configuration that allows a client application to use Stronghold as an OIDC provider.

1. Enable an Stronghold auth method:

   ```
   $ d8 stronghold auth enable userpass
   Success! Enabled userpass auth method at: userpass/
   ```

   Any Stronghold auth method may be used within the OIDC flow. For simplicity, enable the `userpass` auth method.

2. Create a user:

   ```
   $ d8 stronghold write auth/userpass/users/end-user password="securepassword"
   Success! Data written to: auth/userpass/users/end-user
   ```

   This user will authenticate to Stronghold through a client application, otherwise known as an OIDC relying party.

3. Create a client application:

   ```
   $ d8 stronghold write identity/oidc/client/my-webapp \
     redirect_uris="https://localhost:9702/auth/oidc-callback" \
     assignments="allow_all"
   Success! Data written to: identity/oidc/client/my-webapp
   ```

   This operation creates a client application which can be used to configure an OIDC relying party.

The `assignments` parameter limits the Stronghold entities and groups that are allowed to authenticate through the client application. By default, no Stronghold entities are allowed. To allow all Stronghold entities to authenticate, the built-in `allow_all` assignment is provided.

4. Read client credentials:

```
$ d8 stronghold read identity/oidc/client/my-webapp

Key                 Value
---                 -----
access_token_ttl    24h
assignments         [allow_all]
client_id           GSDTnn3KaOrLpNlVGlYLS9TVsZgOTweO
client_secret
hvo_secret_gBKHcTP58C4aq7FqPWsuqKgpiiegd7ahpifGae9WGkHRCwFEJTZA9KGdNVpzE0r8
client_type         confidential
id_token_ttl        24h
key                 default
redirect_uris       [https://localhost:9702/auth/oidc-callback]
```

The `client_id` and `client_secret` are the client application's credentials. These values are typically required when configuring an OIDC relying party.

5. Read OIDC discovery configuration:

```
$ curl -s http://127.0.0.1:8200/v1/identity/oidc/provider/default/.well-known/
openid-configuration
{
  "issuer": "http://127.0.0.1:8200/v1/identity/oidc/provider/default",
  "jwks_uri": "http://127.0.0.1:8200/v1/identity/oidc/provider/default/.well-
known/keys",
  "authorization_endpoint": "http://127.0.0.1:8200/ui/vault/identity/oidc/
provider/default/authorize",
  "token_endpoint": "http://127.0.0.1:8200/v1/identity/oidc/provider/default/
token",
  "userinfo_endpoint": "http://127.0.0.1:8200/v1/identity/oidc/provider/default/
userinfo",
  "request_parameter_supported": false,
  "request_uri_parameter_supported": false,
  "id_token_signing_alg_values_supported": [
    "RS256",
    "RS384",
    "RS512",
    "ES256",
    "ES384",
    "ES512",
    "EdDSA"
  ],
  "response_types_supported": [
    "code"
  ],
  "scopes_supported": [
    "openid"
  ],
  "subject_types_supported": [
    "public"
  ],
  "grant_types_supported": [
    "authorization_code"
  ],
  "token_endpoint_auth_methods_supported": [
    "none",
    "client_secret_basic",
    "client_secret_post"
  ]
}
```

Each Stronghold OIDC provider publishes discovery metadata. The `issuer` value is typically required when configuring an OIDC relying party.

Usage

After configuring an Stronghold auth method and client application, the following details can be used to configure an OIDC relying party to delegate end-user authentication to Stronghold.

- `client_id` - The ID of the client application
- `client_secret` - The secret of the client application
- `issuer` - The issuer of the Stronghold OIDC provider

Otherwise, refer to the documentation of the specific OIDC relying party for usage details.

Supported flows

The Stronghold OIDC provider feature currently supports the following authentication flow:

- Authorization Code Flow.

OIDC identity tokens

Introduction

Identity information is used throughout Stronghold, but it can also be exported for use by other applications. An authorized user/application can request a token that encapsulates identity information for their associated entity. These tokens are signed JWTs following the OIDC ID token structure. The public keys used to authenticate the tokens are published by Stronghold on an unauthenticated endpoint following OIDC discovery and JWKS conventions, which should be directly usable by JWT/OIDC libraries. An introspection endpoint is also provided by Stronghold for token verification.

Roles and keys

OIDC-compliant ID tokens are generated against a role which allows configuration of token claims via a templating system, token ttl, and a way to specify which “key” will be used to sign the token. The role template is an optional parameter to customize the token contents and is described in the next section. Token TTL controls the expiration time of the token, after which verification libraries will consider the token invalid. All roles have an associated `client_id` that will be added to the token’s `aud` parameter. JWT/OIDC libraries will usually require this value. The parameter may be set by the operator to a chosen value, or a Stronghold-generated value will be used if left unconfigured.

A role’s `key` parameter links a role to an existing named key (multiple roles may refer to the same key). It is not possible to generate an unsigned ID token.

A named key is a public/private key pair generated by Stronghold. The private key is used to sign the identity tokens, and the public key is used by clients to verify the signature. Keys are regularly rotated, whereby a new key pair is generated and the previous *public* key is retained for a limited time for verification purposes.

A named key’s configuration specifies a rotation period, a verification ttl, signing algorithm and allowed client IDs. Rotation period specifies the frequency at which a new signing key is generated and the private portion of the previous signing key is deleted. Verification ttl is the time a public key is retained for

verification after being rotated. By default, keys are rotated every 24 hours, and continue to be available for verification for 24 hours after their rotation.

A key's list of allowed client IDs limits which roles may reference the key. The parameter may be set to `*` to allow all roles. The validity evaluation is made when a token is requested, not during configuration.

Token contents and templates

Identity tokens will always contain, at a minimum, the claims required by OIDC:

- `iss` - Issuer URL
- `sub` - Requester's entity ID
- `aud` - `client_id` for the role
- `iat` - Time of issue
- `exp` - Expiration time for the token

In addition, the operator may configure per-role templates that allow a variety of other entity information to be added to the token. The templates are structured as JSON with replaceable parameters. The parameter syntax is the same as that used for ACL Path Templating.

For example:

```
{
  "color": {{identity.entity.metadata.color}},
  "userinfo": {
     "username": {{identity.entity.aliases.usermap_123.metadata.username}},
     "groups": {{identity.entity.groups.names}}
  },
  "nbf": {{time.now}}
}
```

When a token is requested, the resulting template might be populated as:

```
{
  "color": "green",
  "userinfo": {
     "username": "bob",
     "groups": ["web", "engr", "default"]
  },
  "nbf": 1561411915
}
```

which would be merged with the base OIDC claims into the final token:

```
{
  "iss": "https://10.1.1.45:8200/v1/identity/oidc",
  "sub": "a2cd63d3-5364-406f-980e-8d71bb0692f5",
  "aud": "SxSouteCYPBoaTFy94hFghmekos",
  "iat": 1561411915,
  "exp": 1561412215,
  "color": "green",
  "userinfo": {
    "username": "bob",
    "groups": ["web", "engr", "default"]
  },
  "nbf": 1561411915
}
```

Note how the template is merged, with top level template keys becoming top level token keys. For this reason, templates may not contain top level keys that overwrite the standard OIDC claims.

Template parameters that are not present for an entity, such as a metadata that isn’t present, or an alias accessor which doesn’t exist, are simply empty strings or objects, depending on the data type.

Templates are configured on the role and may be optionally encoded as base64.

The full list of template parameters is shown below:

Name	Description
`identity.entity.id`	The entity’s ID
`identity.entity.name`	The entity’s name
`identity.entity.groups.ids`	The IDs of the groups the entity is a member of
`identity.entity.groups.names`	The names of the groups the entity is a member of
`identity.entity.metadata`	Metadata associated with the entity
`identity.entity.metadata.<metadata key>`	Metadata associated with the entity for the given key
`identity.entity.aliases.<mount accessor>.id`	Entity alias ID for the given mount
`identity.entity.aliases.<mount accessor>.name`	Entity alias name for the given mount
`identity.entity.aliases.<mount accessor>.metadata`	Metadata associated with the alias for the given mount
`identity.entity.aliases.<mount accessor>.metadata.<metadata key>`	Metadata associated with the alias for the given mount and metadata key
`identity.entity.aliases.<mount accessor>.custom_metadata`	Custom metadata associated with the alias for the given mount
`identity.entity.aliases.<mount accessor>.custom_metadata.<custom_metadata key>`	Custom metadata associated with the alias for the given mount and custom metadata key
`time.now`	Current time as integral seconds since the Epoch
`time.now.plus.<duration>`	Current time plus a duration format string
`time.now.minus.<duration>`	Current time minus a duration format string

Token generation

An authenticated client may request a token using the token generation endpoint. The token will be generated per the requested role's specifications, for the requester's entity. It is not possible to generate tokens for an arbitrary entity.

Verifying authenticity of ID tokens generated by Stronghold

An identity token may be verified by the client party using the public keys published by Stronghold, or via an Stronghold-provided introspection endpoint.

Stronghold will serve standard ".well-known" endpoints that allow easy integration with OIDC verification libraries. Configuring the libraries will typically involve providing an issuer URL and client ID. The library will then handle key requests and can validate the signature and claims requirements on tokens. This approach has the advantage of only requiring *access* to Stronghold, not *authorization*, as the .well-known endpoints are unauthenticated.

Alternatively, the token may be sent to Stronghold for verification via an `introspection endpoint`. The response will indicate whether the token is "active" or not, as well as any errors that occurred during validation. Beyond simply allowing the client to delegate verification to Stronghold, using this endpoint incorporates the additional check of whether the entity is still active or not, which is something that cannot be determined from the token alone. Unlike the .well-known endpoint, accessing the introspection endpoint does require a valid Stronghold token and sufficient authorization.

Issuer considerations

The identity token system has one configurable parameter: issuer. The issuer `iss` claim is particularly important for proper validation of the token by clients. Consumers of the token will request public keys from Stronghold using the issuer URL, so it must be network reachable. Furthermore, the returned set of keys will include an issuer that must match the request.

By default Stronghold will set the issuer to the Stronghold instance's `api_addr`. This means that tokens issued in a given cluster should be validated within that same cluster. Alternatively, the `issuer` parameter may be configured explicitly. This address must point to the identity/oidc path for the Stronghold instance (e.g. `https://stronghold.example.com:8200/v1/identity/oidc`) and should be reachable by any client trying to validate identity tokens.

Databases Secrets engine

Overview

Databases

The database secrets engine generates database credentials dynamically based on configured roles. It works with a number of different databases through a plugin interface. There are a number of built-in database types, and an exposed framework for running custom database types for extendability. This means that services that need to access a database no longer need to hardcode credentials: they can

request them from Stronghold, and use Stronghold's leasing mechanism to more easily roll keys. These are referred to as "dynamic roles" or "dynamic secrets".

Since every service is accessing the database with unique credentials, it makes auditing much easier when questionable data access is discovered. You can track it down to the specific instance of a service based on the SQL username.

Stronghold makes use of its own internal revocation system to ensure that users become invalid within a reasonable time of the lease expiring.

Static roles

With dynamic secrets, Stronghold generates a unique username and password pair for each unique credential request. Stronghold also supports **static roles** for some database secrets engines. Static roles are a 1-to-1 mapping of Stronghold roles to usernames in a database. With static roles, Stronghold stores, and automatically rotates, passwords for the associated database user based on a configurable period of time.

When a client requests credentials for the static role, Stronghold returns the current password for whichever database user is mapped to the requested role. With static roles, anyone with the proper Stronghold policies can access the associated user account in the database.

> ⚠ [Do not use static roles for root database credentials] Do not manage the same root database credentials that you provide to Stronghold in `config/` with static roles.
> Stronghold does not distinguish between standard credentials and root credentials when rotating passwords. If you assign your root credentials to a static role, any dynamic or static users managed by that database configuration will fail after rotation because the password for `config/` is no longer valid.
> If you need to rotate root credentials, use the `rotate-root-credentials` API endpoint.

Refer to the database capabilities table to determine if your chosen database backend supports static roles.

Setup

Most secrets engines must be configured in advance before they can perform their functions. These steps are usually completed by an operator or configuration management tool.

1. Enable the database secrets engine:

   ```
   $ d8 stronghold secrets enable database
   Success! Enabled the database secrets engine at: database/
   ```

   By default, the secrets engine will enable at the name of the engine. To enable the secrets engine at a different path, use the `-path` argument.

2. Configure Stronghold with the proper plugin and connection information:

```
$ d8 stronghold write database/config/my-database \
    plugin_name="..." \
    connection_url="..." \
    allowed_roles="..." \
    username="..." \
    password="..." \
```

> ⚠ It is highly recommended a user within the database is created specifically for Stronghold to use. This user will be used to manipulate dynamic and static users within the database. This user is called the “root” user within the documentation.

Stronghold will use the user specified here to create/update/revoke database credentials. That user must have the appropriate permissions to perform actions upon other database users (create, update credentials, delete, etc.).

This secrets engine can configure multiple database connections. For details on the specific configuration options, please see the database-specific documentation.

1. After configuring the root user, it is highly recommended you rotate that user’s password such that the stronghold user is not accessible by any users other than Stronghold itself:

   ```
   d8 stronghold write -force database/rotate-root/my-database
   ```

> ℹ When this is done, the password for the user specified in the previous step is no longer accessible. Because of this, it is highly recommended that a user is created specifically for Stronghold to use to manage database users.

1. Configure a role that maps a name in Stronghold to a set of creation statements to create the database credential:

   ```
   $ d8 stronghold write database/roles/my-role \
       db_name=my-database \
       creation_statements="..." \
       default_ttl="1h" \
       max_ttl="24h"
   Success! Data written to: database/roles/my-role
   ```

   The `{{username}}` and `{{password}}` fields will be populated by the plugin with dynamically generated values. In some plugins the `{{expiration}}` field is also supported.

Usage

After the secrets engine is configured and a user/machine has an Stronghold token with the proper permission, it can generate credentials.

1. Generate a new credential by reading from the `/creds` endpoint with the name of the role:

```
$ d8 stronghold read database/creds/my-role
Key                Value
---                -----
lease_id           database/creds/my-role/2f6a614c-4aa2-7b19-24b9-ad944a8d4de6
lease_duration     1h
lease_renewable    true
password           FSREZ1S0kFsZtLat-y94
username           v-strongholduser-e2978cd0-ugp7iqI2hdlff5hfjylJ-1602537260
```

Database capabilities

All databases support dynamic roles and static roles. All plugins support rotating the root user's credentials.

Database	Root Credential Rotation	Dynamic Roles	Static Roles	Username Customization	Credential Types
MySQL/ MariaDB	Yes	Yes	Yes	Yes	password
PostgreSQL	Yes	Yes	Yes	Yes	password

Credential types

Database systems support a variety of authentication methods and credential types. The database secrets engine supports management of credentials alternative to usernames and passwords. The `credential_type` and `credential_config` parameters of dynamic and static roles configure the credential that Stronghold will generate and make available to database plugins. See the documentation of individual database plugins for the credential types they support and usage examples.

Password generation

Passwords are generated via password policies. Databases can optionally set a password policy for use across all roles or at the individual role level for that database. For example, each time you call `d8 stronghold write database/config/my-database` you can specify a password policy for all roles using `my-database`. Each database has a default password policy defined as: 20 characters with at least 1 uppercase character, at least 1 lowercase character, at least 1 number, and at least 1 dash character.

The default password generation can be represented as the following password policy:

```
length = 20

rule "charset" {
 charset = "abcdefghijklmnopqrstuvwxyz"
 min-chars = 1
}
rule "charset" {
 charset = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
 min-chars = 1
}
rule "charset" {
 charset = "0123456789"
 min-chars = 1
}
rule "charset" {
 charset = "-"
 min-chars = 1
}
```

Disable character escaping

You can specify the option `disable_escaping` with a value of `true` in some secrets engines to prevent Stronghold from escaping special characters in the username and password fields. This is necessary for some alternate connection string formats.

For example, when the password contains URL-escaped characters like `#` or `%` they will remain as so instead of becoming `%23` and `%25` respectively.

```
$ d8 stronghold write database/config/my-mysql-database \
plugin_name="mysql-database-plugin" \
connection_url='server=localhost;port=3306;user
id={{username}};password={{password}};database=mydb;' \
username="root" \
password='your#StrongPassword%' \
disable_escaping="true"
```

PostgreSQL

PostgreSQL database secrets engine

PostgreSQL is one of the supported plugins for the database secrets engine. This plugin generates database credentials dynamically based on configured roles for the PostgreSQL database, and also supports Static Roles.

Capabilities

Plugin Name	Root Credential Rotation	Dynamic Roles	Static Roles	Username Customization
`postgresql-database-plugin`	Yes	Yes	Yes	Yes (1.7+)

Setup

1. Enable the database secrets engine if it is not already enabled:

   ```
   $ d8 stronghold secrets enable database
   Success! Enabled the database secrets engine at: database/
   ```

   By default, the secrets engine will enable at the name of the engine. To enable the secrets engine at a different path, use the `-path` argument.

2. Configure Stronghold with the proper plugin and connection information:

   ```
   $ d8 stronghold write database/config/my-postgresql-database \
       plugin_name="postgresql-database-plugin" \
       allowed_roles="my-role" \
       connection_url="postgresql://{{username}}:{{password}}@localhost:5432/
   database-name" \
       username="strongholduser" \
       password="strongholdpass" \
       password_authentication="scram-sha-256"
   ```

3. Configure a role that maps a name in Stronghold to an SQL statement to execute to create the database credential:

   ```
   $ d8 stronghold write database/roles/my-role \
       db_name="my-postgresql-database" \
       creation_statements="CREATE ROLE \"{{name}}\" WITH LOGIN PASSWORD
   '{{password}}' VALID UNTIL '{{expiration}}'; \
           GRANT SELECT ON ALL TABLES IN SCHEMA public TO \"{{name}}\";" \
       default_ttl="1h" \
       max_ttl="24h"
   Success! Data written to: database/roles/my-role
   ```

Usage

After the secrets engine is configured and a user/machine has an Stronghold token with the proper permission, it can generate credentials.

1. Generate a new credential by reading from the `/creds` endpoint with the name of the role:

```
$ d8 stronghold read database/creds/my-role
Key                Value
---                -----
lease_id           database/creds/my-role/2f6a614c-4aa2-7b19-24b9-ad944a8d4de6
lease_duration     1h
lease_renewable    true
password           SsnoaA-8Tv4t34f41baD
username           v-strongholduse-my-role-x
```

MySQL

MySQL/MariaDB database secrets engine

MySQL is one of the supported plugins for the database secrets engine. This plugin generates database credentials dynamically based on configured roles for the MySQL database, and also supports Static Roles.

This plugin has a few different instances built into Stronghold, each instance is for a slightly different MySQL driver. The only difference between these plugins is the length of usernames generated by the plugin as different versions of mysql accept different lengths. The available plugins are:

- mysql-database-plugin
- mysql-aurora-database-plugin
- mysql-rds-database-plugin
- mysql-legacy-database-plugin

Capabilities

Plugin Name	Root Credential Rotation	Dynamic Roles	Static Roles	Username Customization
Can vary	Yes	Yes	Yes	Yes

Setup

1. Enable the database secrets engine if it is not already enabled:

   ```
   $ d8 stronghold secrets enable database
   Success! Enabled the database secrets engine at: database/
   ```

   By default, the secrets engine will enable at the name of the engine. To enable the secrets engine at a different path, use the `-path` argument.

2. Configure Stronghold with the proper plugin and connection information:

```
$ d8 stronghold write database/config/my-mysql-database \
    plugin_name=mysql-database-plugin \
    connection_url="{{username}}:{{password}}@tcp(127.0.0.1:3306)/" \
    allowed_roles="my-role" \
    username="strongholduser" \
    password="strongholdpass"
```

3. Configure a role that maps a name in Stronghold to an SQL statement to execute to create the database credential:

```
$ d8 stronghold write database/roles/my-role \
    db_name=my-mysql-database \
    creation_statements="CREATE USER '{{name}}'@'%' IDENTIFIED BY
'{{password}}';GRANT SELECT ON *.* TO '{{name}}'@'%';" \
    default_ttl="1h" \
    max_ttl="24h"
Success! Data written to: database/roles/my-role
```

Usage

After the secrets engine is configured and a user/machine has an Stronghold token with the proper permission, it can generate credentials.

1. Generate a new credential by reading from the `/creds` endpoint with the name of the role:

```
$ d8 stronghold read database/creds/my-role
Key                Value
---                -----
lease_id           database/creds/my-role/2f6a614c-4aa2-7b19-24b9-ad944a8d4de6
lease_duration     1h
lease_renewable    true
password           yY-57n3X5UQhxnmFRP3f
username           v_strongholduser_my-role_crBWVqVh2Hc1
```

Client x509 certificate authentication

This plugin supports using MySQL's x509 Client-side Certificate Authentication

To use this authentication mechanism, configure the plugin:

```
$ d8 stronghold write database/config/my-mysql-database \
    plugin_name=mysql-database-plugin \
    allowed_roles="my-role" \
    connection_url="user:password@tcp(localhost:3306)/test" \
    tls_certificate_key=@/path/to/client.pem \
    tls_ca=@/path/to/client.ca
```

Note: `tls_certificate_key` and `tls_ca` map to `ssl-cert (combined with ssl-key)` and `ssl-ca` configuration options from MySQL with the exception that the Stronghold parameters are the contents of those files, not filenames. As such, the two options are independent of each other. See the MySQL Connection Options for more information.

Examples

Using wildcards in grant statements

MySQL supports using wildcards in grant statements. These are sometimes needed by applications which expect access to a large number of databases inside MySQL. This can be realized by using a wildcard in the grant statement. For example if you want the user created by Stronghold to have access to all databases starting with `fooapp_` you could use the following creation statement:

```
CREATE USER '{{name}}'@'%' IDENTIFIED BY '{{password}}'; GRANT SELECT ON `fooapp\_%`.*
TO '{{name}}'@'%';
```

MySQL expects the part in which the wildcards are to be placed inside backticks. If you want to add this creation statement to Stronghold via the Stronghold CLI you cannot simply paste the above statement on the CLI because the shell will interpret the text between the backticks as something that must be executed. The easiest way to get around this is to encode the creation statement as Base64 and feed this to Stronghold. For example:

```
$ d8 stronghold write database/roles/my-role \
    db_name=mysql \

creation_statements="Q1JFQVRFIFVTRVIgJ3t7bmFtZX19J0AnJScgSURFTlRJRklFRCBCWSAne3twYXNzd
29yZH19JzsgR1JBTlQgU0VMRUNUIE9OIGBmb29hcHBcXyVgLiogVE8gJ3t7bmFtZX19J0AnJSc7" \
    default_ttl="1h" \
    max_ttl="24h"
```

Rotating root credentials in MySQL 5.6

The default root rotation setup for MySQL uses the `ALTER USER` syntax present in MySQL 5.7 and up. For MySQL 5.6, `root_rotation_statements` must be configured to use the old `SET PASSWORD` syntax. For example:

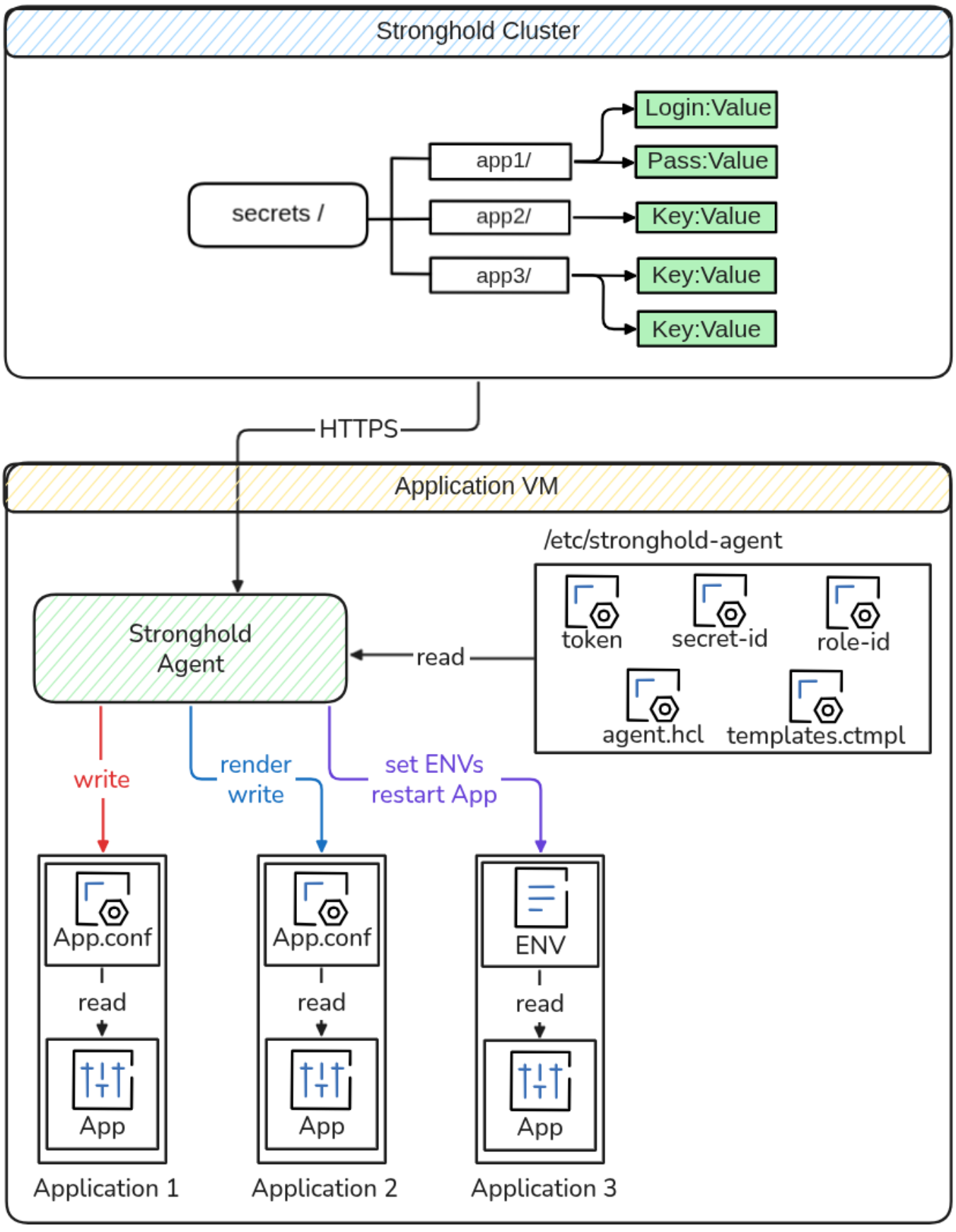
```
$ d8 stronghold write database/config/my-mysql-database \
    plugin_name=mysql-database-plugin \
    connection_url="{{username}}:{{password}}@tcp(127.0.0.1:3306)/" \
    root_rotation_statements="SET PASSWORD = PASSWORD('{{password}}')" \
    allowed_roles="my-role" \
    username="root" \
    password="mysql"
```

Stronghold agent

Overview

coming soon…

Functional scheme



Use cases

coming soon…

Key features

coming soon…

Main settings

coming soon…

Launch and management

coming soon…

Reference

API

auth

auth

GET /auth/token/accessors

Operation ID: `token-list-accessors`

List token accessors, which can then be be used to iterate and discover their properties or revoke them. Because this can be used to cause a denial of service, this endpoint requires ‘sudo’ capability in addition to ’list’.

Required sudo: yes

Parameters

Parameter	Type	Location	Required	Description
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

POST /auth/token/create

Operation ID: `token-create`

The token create path is used to create new tokens.

Request body parameters

Parameter	Type	Required	Description
`display_name`	string	no	Name to associate with this token
`entity_alias`	string	no	Name of the entity alias to associate with this token
`explicit_max_ttl`	string	no	Explicit Max TTL of this token
`id`	string	no	Value for the token
`lease`	string	no	⚠ Deprecated. Use 'ttl' instead
`meta`	object	no	Arbitrary key=value metadata to associate with the token
`no_default_policy`	boolean	no	Do not include default policy for this token
`no_parent`	boolean	no	Create the token with no parent
`num_uses`	integer	no	Max number of uses for this token
`period`	string	no	Renew period
`policies`	array	no	List of policies for the token
`renewable`	boolean (default: True)	no	Allow token to be renewed past its initial TTL up to system/mount maximum TTL
`ttl`	string	no	Time to live for this token
`type`	string	no	Token type

Responses

200: OK

POST /auth/token/create-orphan

Operation ID: `token-create-orphan`

The token create path is used to create new orphan tokens.

Request body parameters

Parameter	Type	Required	Description
`display_name`	string	no	Name to associate with this token
`entity_alias`	string	no	Name of the entity alias to associate with this token
`explicit_max_ttl`	string	no	Explicit Max TTL of this token
`id`	string	no	Value for the token
`lease`	string	no	⚠ Deprecated. Use 'ttl' instead
`meta`	object	no	Arbitrary key=value metadata to associate with the token
`no_default_policy`	boolean	no	Do not include default policy for this token
`no_parent`	boolean	no	Create the token with no parent
`num_uses`	integer	no	Max number of uses for this token
`period`	string	no	Renew period
`policies`	array	no	List of policies for the token
`renewable`	boolean (default: True)	no	Allow token to be renewed past its initial TTL up to system/mount maximum TTL
`ttl`	string	no	Time to live for this token
`type`	string	no	Token type

Responses

200: OK

POST /auth/token/create/{role_name}

Operation ID: `token-create-against-role`

This token create path is used to create new tokens adhering to the given role.

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role

Request body parameters

Parameter	Type	Required	Description
`display_name`	string	no	Name to associate with this token
`entity_alias`	string	no	Name of the entity alias to associate with this token
`explicit_max_ttl`	string	no	Explicit Max TTL of this token
`id`	string	no	Value for the token
`lease`	string	no	⚠ Deprecated. Use 'ttl' instead
`meta`	object	no	Arbitrary key=value metadata to associate with the token
`no_default_policy`	boolean	no	Do not include default policy for this token
`no_parent`	boolean	no	Create the token with no parent
`num_uses`	integer	no	Max number of uses for this token
`period`	string	no	Renew period
`policies`	array	no	List of policies for the token
`renewable`	boolean (default: True)	no	Allow token to be renewed past its initial TTL up to system/mount maximum TTL
`ttl`	string	no	Time to live for this token
`type`	string	no	Token type

Responses

200: OK

GET /auth/token/lookup

Operation ID: `token-look-up-2`

This endpoint will lookup a token and its properties.

Responses

200: OK

POST /auth/token/lookup

Operation ID: `token-look-up`

This endpoint will lookup a token and its properties.

Request body parameters

Parameter	Type	Required	Description
`token`	string	no	Token to lookup (POST request body)

Responses

200: OK

POST /auth/token/lookup-accessor

Operation ID: `token-look-up-accessor`

This endpoint will lookup a token associated with the given accessor and its properties. Response will not contain the token ID.

Request body parameters

Parameter	Type	Required	Description
`accessor`	string	no	Accessor of the token to look up (request body)

Responses

200: OK

GET /auth/token/lookup-self

Operation ID: `token-look-up-self`

This endpoint will lookup a token and its properties.

Responses

200: OK

POST /auth/token/lookup-self

Operation ID: `token-look-up-self2`

This endpoint will lookup a token and its properties.

Request body parameters

Parameter	Type	Required	Description
`token`	string	no	Token to look up (unused, does not need to be set)

Responses

200: OK

POST /auth/token/renew

Operation ID: `token-renew`

This endpoint will renew the given token and prevent expiration.

Request body parameters

Parameter	Type	Required	Description
`increment`	integer (default: 0)	no	The desired increment in seconds to the token expiration
`token`	string	no	Token to renew (request body)

Responses

200: OK

POST /auth/token/renew-accessor

Operation ID: `token-renew-accessor`

This endpoint will renew a token associated with the given accessor and its properties. Response will not contain the token ID.

Request body parameters

Parameter	Type	Required	Description
`accessor`	string	no	Accessor of the token to renew (request body)
`increment`	integer (default: 0)	no	The desired increment in seconds to the token expiration

Responses

200: OK

POST /auth/token/renew-self

Operation ID: `token-renew-self`

This endpoint will renew the token used to call it and prevent expiration.

Request body parameters

Parameter	Type	Required	Description
`increment`	integer (default: 0)	no	The desired increment in seconds to the token expiration
`token`	string	no	Token to renew (unused, does not need to be set)

Responses

200: OK

POST /auth/token/revoke

Operation ID: `token-revoke`

This endpoint will delete the given token and all of its child tokens.

Request body parameters

Parameter	Type	Required	Description
`token`	string	no	Token to revoke (request body)

Responses

200: OK

POST /auth/token/revoke-accessor

Operation ID: `token-revoke-accessor`

This endpoint will delete the token associated with the accessor and all of its child tokens.

Request body parameters

Parameter	Type	Required	Description
`accessor`	string	no	Accessor of the token (request body)

Responses

200: OK

POST /auth/token/revoke-orphan

Operation ID: `token-revoke-orphan`

This endpoint will delete the token and orphan its child tokens.

Request body parameters

Parameter	Type	Required	Description
`token`	string	no	Token to revoke (request body)

Responses

200: OK

POST /auth/token/revoke-self

Operation ID: `token-revoke-self`

This endpoint will delete the token used to call it and all of its child tokens.

Responses

200: OK

GET /auth/token/roles

Operation ID: `token-list-roles`

This endpoint lists configured roles.

Parameters

Parameter	Type	Location	Required	Description
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /auth/token/roles/{role_name}

Operation ID: `token-read-role`

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role

Responses

200: OK

POST /auth/token/roles/{role_name}

Operation ID: `token-write-role`

Creation supported: yes

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role

Request body parameters

Parameter	Type	Required	Description
`allowed_entity_aliases`	array	no	String or JSON list of allowed entity aliases. If set, specifies the entity aliases which are allowed to be used during token generation. This field supports globbing.
`allowed_policies`	array	no	If set, tokens can be created with any subset of the policies in this list, rather than the normal semantics of tokens being a subset of the calling token's policies. The parameter is a comma-delimited string of policy names.
`allowed_policies_glob`	array	no	If set, tokens can be created with any subset of glob matched policies in this list, rather than the normal semantics of tokens being a subset of the calling token's policies. The parameter is a comma-delimited string of policy name globs.
`bound_cidrs`	array	no	⚠ Deprecated. Use 'token_bound_cidrs' instead.
`disallowed_policies`	array	no	If set, successful token creation via this role will require that no policies in the given list are requested. The parameter is a comma-delimited string of policy names.
`disallowed_policies_glob`	array	no	If set, successful token creation via this role will require that no requested policies glob match any of policies in this list. The parameter is a comma-delimited string of policy name globs.
`explicit_max_ttl`	integer	no	⚠ Deprecated. Use 'token_explicit_max_ttl' instead.
`orphan`	boolean	no	If true, tokens created via this role will be orphan tokens (have no parent)
`path_suffix`	string	no	If set, tokens created via this role will contain the given suffix as a part of their path. This can be used to assist use of the

Parameter	Type	Required	Description
			‘revoke-prefix’ endpoint later on. The given suffix must match the regular expression.\w[\w-.]+\w
`period`	integer	no	⚠ Deprecated. Use ’token_period’ instead.
`renewable`	boolean (default: True)	no	Tokens created via this role will be renewable or not according to this value. Defaults to “true”.
`token_bound_cidrs`	array	no	Comma separated string or JSON list of CIDR blocks. If set, specifies the blocks of IP addresses which are allowed to use the generated token.
`token_explicit_max_ttl`	integer	no	If set, tokens created via this role carry an explicit maximum TTL. During renewal, the current maximum TTL values of the role and the mount are not checked for changes, and any updates to these values will have no effect on the token being renewed.
`token_no_default_policy`	boolean	no	If true, the ‘default’ policy will not automatically be added to generated tokens
`token_num_uses`	integer	no	The maximum number of times a token may be used, a value of zero means unlimited
`token_period`	integer	no	If set, tokens created via this role will have no max lifetime; instead, their renewal period will be fixed to this value. This takes an integer number of seconds, or a string duration (e.g. “24h”).
`token_type`	string (default: default-service)	no	The type of token to generate, service or batch

Responses

200: OK

DELETE /auth/token/roles/{role_name}

Operation ID: `token-delete-role`

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role

Responses

204: empty body

POST /auth/token/tidy

Operation ID: `token-tidy`

This endpoint performs cleanup tasks that can be run if certain error conditions have occurred.

Responses

200: OK

POST /auth/{approle_mount_path}/login

Operation ID: `app-role-login`

Issue a token based on the credentials supplied

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`role_id`	string	no	Unique identifier of the Role. Required to be supplied when the ‘bind_secret_id’ constraint is set.
`secret_id`	string (default:)	no	SecretID belong to the App role

Responses

200: OK

GET /auth/{approle_mount_path}/role

Operation ID: `app-role-list-roles`

Lists all the roles registered with the backend.

Parameters

Parameter	Type	Location	Required	Description
`approle_mount_path`	string	path	yes	Path that the backend was mounted at
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

Parameter	Type	Required	Description
`keys`	array	no	

GET /auth/{approle_mount_path}/role/{role_name}

Operation ID: `app-role-read-role`

Register an role with the backend.

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`bind_secret_id`	boolean	no	Impose secret ID to be presented when logging in using this role.
`local_secret_ids`	boolean	no	If true, the secret identifiers generated using this role will be cluster local. This can only be set during role creation and once set, it can’t be reset later
`period`	integer	no	⚠ Deprecated. Use “token_period” instead. If this and “token_period” are both specified, only “token_period” will be used.
`policies`	array	no	⚠ Deprecated. Use “token_policies” instead. If this and “token_policies” are both specified, only “token_policies” will be used.
`secret_id_bound_cidrs`	array	no	Comma separated string or list of CIDR blocks. If set, specifies the blocks of IP addresses which can perform the login operation.
`secret_id_num_uses`	integer	no	Number of times a secret ID can access the role, after which the secret ID will expire.
`secret_id_ttl`	integer	no	Duration in seconds after which the issued secret ID expires.
`token_bound_cidrs`	array	no	Comma separated string or JSON list of CIDR blocks. If set, specifies the blocks of IP addresses which are allowed to use the generated token.
`token_explicit_max_ttl`	integer	no	If set, tokens created via this role carry an explicit maximum TTL. During renewal, the current maximum TTL values of the role and the mount are not checked for changes, and any updates to these values will have no effect on the token being renewed.
`token_max_ttl`	integer	no	

Parameter	Type	Required	Description
			The maximum lifetime of the generated token
`token_no_default_policy`	boolean	no	If true, the ‘default’ policy will not automatically be added to generated tokens
`token_num_uses`	integer	no	The maximum number of times a token may be used, a value of zero means unlimited
`token_period`	integer	no	If set, tokens created via this role will have no max lifetime; instead, their renewal period will be fixed to this value.
`token_policies`	array	no	Comma-separated list of policies
`token_ttl`	integer	no	The initial ttl of the token to generate
`token_type`	string (default: default-service)	no	The type of token to generate, service or batch

POST /auth/{approle_mount_path}/role/{role_name}

Operation ID: `app-role-write-role`

Register an role with the backend.

Creation supported: yes

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`bind_secret_id`	boolean (default: True)	no	Impose secret_id to be presented when logging in using this role. Defaults to 'true'.
`bound_cidr_list`	array	no	⚠ Deprecated. Use "secret_id_bound_cidrs" instead.
`local_secret_ids`	boolean	no	If set, the secret IDs generated using this role will be cluster local. This can only be set during role creation and once set, it can't be reset later.
`period`	integer	no	⚠ Deprecated. Use "token_period" instead. If this and "token_period" are both specified, only "token_period" will be used.
`policies`	array	no	⚠ Deprecated. Use "token_policies" instead. If this and "token_policies" are both specified, only "token_policies" will be used.
`role_id`	string	no	Identifier of the role. Defaults to a UUID.
`secret_id_bound_cidrs`	array	no	Comma separated string or list of CIDR blocks. If set, specifies the blocks of IP addresses which can perform the login operation.
`secret_id_num_uses`	integer	no	Number of times a SecretID can access the role, after which the SecretID will expire. Defaults to 0 meaning that the the secret_id is of unlimited use.
`secret_id_ttl`	integer	no	Duration in seconds after which the issued SecretID should expire. Defaults to 0, meaning no expiration.
`token_bound_cidrs`	array	no	Comma separated string or JSON list of CIDR blocks. If set, specifies the blocks of IP addresses which are allowed to use the generated token.
`token_explicit_max_ttl`	integer	no	If set, tokens created via this role carry an explicit maximum TTL. During renewal, the

Parameter	Type	Required	Description
			current maximum TTL values of the role and the mount are not checked for changes, and any updates to these values will have no effect on the token being renewed.
`token_max_ttl`	integer	no	The maximum lifetime of the generated token
`token_no_default_policy`	boolean	no	If true, the ‘default’ policy will not automatically be added to generated tokens
`token_num_uses`	integer	no	The maximum number of times a token may be used, a value of zero means unlimited
`token_period`	integer	no	If set, tokens created via this role will have no max lifetime; instead, their renewal period will be fixed to this value. This takes an integer number of seconds, or a string duration (e.g. “24h”).
`token_policies`	array	no	Comma-separated list of policies
`token_ttl`	integer	no	The initial ttl of the token to generate
`token_type`	string (default: default-service)	no	The type of token to generate, service or batch

Responses

200: OK

DELETE /auth/{approle_mount_path}/role/{role_name}

Operation ID: `app-role-delete-role`

Register an role with the backend.

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: No Content

GET /auth/{approle_mount_path}/role/{role_name}/bind-secret-id

Operation ID: `app-role-read-bind-secret-id`

Impose secret_id to be presented during login using this role.

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`bind_secret_id`	boolean	no	Impose secret_id to be presented when logging in using this role. Defaults to 'true'.

POST /auth/{approle_mount_path}/role/{role_name}/bind-secret-id

Operation ID: `app-role-write-bind-secret-id`

Impose secret_id to be presented during login using this role.

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`bind_secret_id`	boolean (default: True)	no	Impose secret_id to be presented when logging in using this role.

Responses

204: No Content

DELETE /auth/{approle_mount_path}/role/{role_name}/bind-secret-id

Operation ID: `app-role-delete-bind-secret-id`

Impose secret_id to be presented during login using this role.

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: No Content

GET /auth/{approle_mount_path}/role/{role_name}/bound-cidr-list

Operation ID: `app-role-read-bound-cidr-list`

Deprecated: Comma separated list of CIDR blocks, if set, specifies blocks of IP addresses which can perform the login operation

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`bound_cidr_list`	array	no	⚠ Deprecated. Deprecated: Please use “secret_id_bound_cidrs” instead. Comma separated string or list of CIDR blocks. If set, specifies the blocks of IP addresses which can perform the login operation.

POST /auth/{approle_mount_path}/role/{role_name}/bound-cidr-list

Operation ID: `app-role-write-bound-cidr-list`

Deprecated: Comma separated list of CIDR blocks, if set, specifies blocks of IP addresses which can perform the login operation

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`bound_cidr_list`	array	no	Deprecated: Please use “secret_id_bound_cidrs” instead. Comma separated string or list of CIDR blocks. If set, specifies the blocks of IP addresses which can perform the login operation.

Responses

204: No Content

DELETE /auth/{approle_mount_path}/role/{role_name}/bound-cidr-list

Operation ID: `app-role-delete-bound-cidr-list`

Deprecated: Comma separated list of CIDR blocks, if set, specifies blocks of IP addresses which can perform the login operation

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: No Content

POST /auth/{approle_mount_path}/role/{role_name}/custom-secret-id

Operation ID: `app-role-write-custom-secret-id`

Assign a SecretID of choice against the role.

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`cidr_list`	array	no	Comma separated string or list of CIDR blocks enforcing secret IDs to be used from specific set of IP addresses. If 'bound_cidr_list' is set on the role, then the list of CIDR blocks listed here should be a subset of the CIDR blocks listed on the role.
`metadata`	string	no	Metadata to be tied to the SecretID. This should be a JSON formatted string containing metadata in key value pairs.
`num_uses`	integer	no	Number of times this SecretID can be used, after which the SecretID expires. Overrides secret_id_num_uses role option when supplied. May not be higher than role's secret_id_num_uses.
`secret_id`	string	no	SecretID to be attached to the role.
`token_bound_cidrs`	array	no	Comma separated string or list of CIDR blocks. If set, specifies the blocks of IP addresses which can use the returned token. Should be a subset of the token CIDR blocks listed on the role, if any.
`ttl`	integer	no	Duration in seconds after which this SecretID expires. Overrides secret_id_ttl role option when supplied. May not be longer than role's secret_id_ttl.

Responses

200: OK

Parameter	Type	Required	Description
`secret_id`	string	no	Secret ID attached to the role.
`secret_id_accessor`	string	no	Accessor of the secret ID
`secret_id_num_uses`	integer	no	Number of times a secret ID can access the role, after which the secret ID will expire.
`secret_id_ttl`	integer	no	Duration in seconds after which the issued secret ID expires.

GET /auth/{approle_mount_path}/role/{role_name}/local-secret-ids

Operation ID: `app-role-read-local-secret-ids`

Enables cluster local secret IDs

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`local_secret_ids`	boolean	no	If true, the secret identifiers generated using this role will be cluster local. This can only be set during role creation and once set, it can’t be reset later

GET /auth/{approle_mount_path}/role/{role_name}/period

Operation ID: `app-role-read-period`

Updates the value of ‘period’ on the role

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`period`	integer	no	⚠ Deprecated. Use “token_period” instead. If this and “token_period” are both specified, only “token_period” will be used.
`token_period`	integer	no	If set, tokens created via this role will have no max lifetime; instead, their renewal period will be fixed to this value. This takes an integer number of seconds, or a string duration (e.g. “24h”).

POST /auth/{approle_mount_path}/role/{role_name}/period

Operation ID: `app-role-write-period`

Updates the value of ‘period’ on the role

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`period`	integer	no	⚠ Deprecated. Use “token_period” instead. If this and “token_period” are both specified, only “token_period” will be used.
`token_period`	integer	no	If set, tokens created via this role will have no max lifetime; instead, their renewal period will be fixed to this value. This takes an integer number of seconds, or a string duration (e.g. “24h”).

Responses

204: No Content

DELETE /auth/{approle_mount_path}/role/{role_name}/period

Operation ID: `app-role-delete-period`

Updates the value of ‘period’ on the role

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: No Content

GET /auth/{approle_mount_path}/role/{role_name}/policies

Operation ID: `app-role-read-policies`

Policies of the role.

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`policies`	array	no	⚠ Deprecated. Use “token_policies” instead. If this and “token_policies” are both specified, only “token_policies” will be used.
`token_policies`	array	no	Comma-separated list of policies

POST /auth/{approle_mount_path}/role/{role_name}/policies

Operation ID: `app-role-write-policies`

Policies of the role.

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`policies`	array	no	⚠ Deprecated. Use “token_policies” instead. If this and “token_policies” are both specified, only “token_policies” will be used.
`token_policies`	array	no	Comma-separated list of policies

Responses

204: No Content

DELETE /auth/{approle_mount_path}/role/{role_name}/policies

Operation ID: `app-role-delete-policies`

Policies of the role.

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: No Content

GET /auth/{approle_mount_path}/role/{role_name}/role-id

Operation ID: `app-role-read-role-id`

Returns the ‘role_id’ of the role.

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`role_id`	string	no	Identifier of the role. Defaults to a UUID.

POST /auth/{approle_mount_path}/role/{role_name}/role-id

Operation ID: `app-role-write-role-id`

Returns the ‘role_id’ of the role.

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`role_id`	string	no	Identifier of the role. Defaults to a UUID.

Responses

204: No Content

GET /auth/{approle_mount_path}/role/{role_name}/secret-id

Operation ID: `app-role-list-secret-ids`

Generate a SecretID against this role.

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

Parameter	Type	Required	Description
`keys`	array	no	

POST /auth/{approle_mount_path}/role/{role_name}/secret-id

Operation ID: `app-role-write-secret-id`

Generate a SecretID against this role.

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`cidr_list`	array	no	Comma separated string or list of CIDR blocks enforcing secret IDs to be used from specific set of IP addresses. If 'bound_cidr_list' is set on the role, then the list of CIDR blocks listed here should be a subset of the CIDR blocks listed on the role.
`metadata`	string	no	Metadata to be tied to the SecretID. This should be a JSON formatted string containing the metadata in key value pairs.
`num_uses`	integer	no	Number of times this SecretID can be used, after which the SecretID expires. Overrides secret_id_num_uses role option when supplied. May not be higher than role's secret_id_num_uses.
`token_bound_cidrs`	array	no	Comma separated string or JSON list of CIDR blocks. If set, specifies the blocks of IP addresses which are allowed to use the generated token.
`ttl`	integer	no	Duration in seconds after which this SecretID expires. Overrides secret_id_ttl role option when supplied. May not be longer than role's secret_id_ttl.

Responses

200: OK

Parameter	Type	Required	Description
`secret_id`	string	no	Secret ID attached to the role.
`secret_id_accessor`	string	no	Accessor of the secret ID
`secret_id_num_uses`	integer	no	Number of times a secret ID can access the role, after which the secret ID will expire.
`secret_id_ttl`	integer	no	Duration in seconds after which the issued secret ID expires.

POST /auth/{approle_mount_path}/role/{role_name}/secret-id-accessor/destroy

Operation ID: `app-role-destroy-secret-id-by-accessor`

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`secret_id_accessor`	string	no	Accessor of the SecretID

Responses

204: No Content

DELETE /auth/{approle_mount_path}/role/{role_name}/secret-id-accessor/destroy

Operation ID: `app-role-destroy-secret-id-by-accessor2`

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: No Content

POST /auth/{approle_mount_path}/role/{role_name}/secret-id-accessor/lookup

Operation ID: `app-role-look-up-secret-id-by-accessor`

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`secret_id_accessor`	string	no	Accessor of the SecretID

Responses

200: OK

Parameter	Type	Required	Description
`cidr_list`	array	no	List of CIDR blocks enforcing secret IDs to be used from specific set of IP addresses. If 'bound_cidr_list' is set on the role, then the list of CIDR blocks listed here should be a subset of the CIDR blocks listed on the role.
`creation_time`	string	no	
`expiration_time`	string	no	
`last_updated_time`	string	no	
`metadata`	object	no	
`secret_id_accessor`	string	no	Accessor of the secret ID
`secret_id_num_uses`	integer	no	Number of times a secret ID can access the role, after which the secret ID will expire.
`secret_id_ttl`	integer	no	Duration in seconds after which the issued secret ID expires.
`token_bound_cidrs`	array	no	List of CIDR blocks. If set, specifies the blocks of IP addresses which can use the returned token. Should be a subset of the token CIDR blocks listed on the role, if any.

GET /auth/{approle_mount_path}/role/{role_name}/secret-id-bound-cidrs

Operation ID: `app-role-read-secret-id-bound-cidrs`

Comma separated list of CIDR blocks, if set, specifies blocks of IP addresses which can perform the login operation

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`secret_id_bound_cidrs`	array	no	Comma separated string or list of CIDR blocks. If set, specifies the blocks of IP addresses which can perform the login operation.

POST /auth/{approle_mount_path}/role/{role_name}/secret-id-bound-cidrs

Operation ID: `app-role-write-secret-id-bound-cidrs`

Comma separated list of CIDR blocks, if set, specifies blocks of IP addresses which can perform the login operation

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`secret_id_bound_cidrs`	array	no	Comma separated string or list of CIDR blocks. If set, specifies the blocks of IP addresses which can perform the login operation.

Responses

204: No Content

DELETE /auth/{approle_mount_path}/role/{role_name}/secret-id-bound-cidrs

Operation ID: `app-role-delete-secret-id-bound-cidrs`

Comma separated list of CIDR blocks, if set, specifies blocks of IP addresses which can perform the login operation

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: No Content

GET /auth/{approle_mount_path}/role/{role_name}/secret-id-num-uses

Operation ID: `app-role-read-secret-id-num-uses`

Use limit of the SecretID generated against the role.

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`secret_id_num_uses`	integer	no	Number of times a secret ID can access the role, after which the SecretID will expire. Defaults to 0 meaning that the secret ID is of unlimited use.

POST /auth/{approle_mount_path}/role/{role_name}/secret-id-num-uses

Operation ID: `app-role-write-secret-id-num-uses`

Use limit of the SecretID generated against the role.

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`secret_id_num_uses`	integer	no	Number of times a SecretID can access the role, after which the SecretID will expire.

Responses

204: No Content

DELETE /auth/{approle_mount_path}/role/{role_name}/secret-id-num-uses

Operation ID: `app-role-delete-secret-id-num-uses`

Use limit of the SecretID generated against the role.

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: No Content

GET /auth/{approle_mount_path}/role/{role_name}/secret-id-ttl

Operation ID: `app-role-read-secret-id-ttl`

Duration in seconds of the SecretID generated against the role.

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`secret_id_ttl`	integer	no	Duration in seconds after which the issued secret ID should expire. Defaults to 0, meaning no expiration.

POST /auth/{approle_mount_path}/role/{role_name}/secret-id-ttl

Operation ID: `app-role-write-secret-id-ttl`

Duration in seconds of the SecretID generated against the role.

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`secret_id_ttl`	integer	no	Duration in seconds after which the issued SecretID should expire. Defaults to 0, meaning no expiration.

Responses

204: No Content

DELETE /auth/{approle_mount_path}/role/{role_name}/secret-id-ttl

Operation ID: `app-role-delete-secret-id-ttl`

Duration in seconds of the SecretID generated against the role.

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: No Content

POST /auth/{approle_mount_path}/role/{role_name}/secret-id/destroy

Operation ID: `app-role-destroy-secret-id`

Invalidate an issued secret_id

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`secret_id`	string	no	SecretID attached to the role.

Responses

204: No Content

DELETE /auth/{approle_mount_path}/role/{role_name}/secret-id/destroy

Operation ID: `app-role-destroy-secret-id2`

Invalidate an issued secret_id

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: No Content

POST /auth/{approle_mount_path}/role/{role_name}/secret-id/lookup

Operation ID: `app-role-look-up-secret-id`

Read the properties of an issued secret_id

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`secret_id`	string	no	SecretID attached to the role.

Responses

200: OK

Parameter	Type	Required	Description
`cidr_list`	array	no	List of CIDR blocks enforcing secret IDs to be used from specific set of IP addresses. If 'bound_cidr_list' is set on the role, then the list of CIDR blocks listed here should be a subset of the CIDR blocks listed on the role.
`creation_time`	string	no	
`expiration_time`	string	no	
`last_updated_time`	string	no	
`metadata`	object	no	
`secret_id_accessor`	string	no	Accessor of the secret ID
`secret_id_num_uses`	integer	no	Number of times a secret ID can access the role, after which the secret ID will expire.
`secret_id_ttl`	integer	no	Duration in seconds after which the issued secret ID expires.
`token_bound_cidrs`	array	no	List of CIDR blocks. If set, specifies the blocks of IP addresses which can use the returned token. Should be a subset of the token CIDR blocks listed on the role, if any.

GET /auth/{approle_mount_path}/role/{role_name}/token-bound-cidrs

Operation ID: `app-role-read-token-bound-cidrs`

Comma separated string or list of CIDR blocks. If set, specifies the blocks of IP addresses which can use the returned token.

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`token_bound_cidrs`	array	no	Comma separated string or list of CIDR blocks. If set, specifies the blocks of IP addresses which can use the returned token. Should be a subset of the token CIDR blocks listed on the role, if any.

POST /auth/{approle_mount_path}/role/{role_name}/token-bound-cidrs

Operation ID: `app-role-write-token-bound-cidrs`

Comma separated string or list of CIDR blocks. If set, specifies the blocks of IP addresses which can use the returned token.

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`token_bound_cidrs`	array	no	Comma separated string or JSON list of CIDR blocks. If set, specifies the blocks of IP addresses which are allowed to use the generated token.

Responses

204: No Content

DELETE /auth/{approle_mount_path}/role/{role_name}/token-bound-cidrs

Operation ID: `app-role-delete-token-bound-cidrs`

Comma separated string or list of CIDR blocks. If set, specifies the blocks of IP addresses which can use the returned token.

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: No Content

GET /auth/{approle_mount_path}/role/{role_name}/token-max-ttl

Operation ID: `app-role-read-token-max-ttl`

Duration in seconds, the maximum lifetime of the tokens issued by using the SecretIDs that were generated against this role, after which the tokens are not allowed to be renewed.

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`token_max_ttl`	integer	no	The maximum lifetime of the generated token

POST /auth/{approle_mount_path}/role/{role_name}/token-max-ttl

Operation ID: `app-role-write-token-max-ttl`

Duration in seconds, the maximum lifetime of the tokens issued by using the SecretIDs that were generated against this role, after which the tokens are not allowed to be renewed.

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`token_max_ttl`	integer	no	The maximum lifetime of the generated token

Responses

204: No Content

DELETE /auth/{approle_mount_path}/role/{role_name}/token-max-ttl

Operation ID: `app-role-delete-token-max-ttl`

Duration in seconds, the maximum lifetime of the tokens issued by using the SecretIDs that were generated against this role, after which the tokens are not allowed to be renewed.

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: No Content

GET /auth/{approle_mount_path}/role/{role_name}/token-num-uses

Operation ID: `app-role-read-token-num-uses`

Number of times issued tokens can be used

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`token_num_uses`	integer	no	The maximum number of times a token may be used, a value of zero means unlimited

POST /auth/{approle_mount_path}/role/{role_name}/token-num-uses

Operation ID: `app-role-write-token-num-uses`

Number of times issued tokens can be used

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`token_num_uses`	integer	no	The maximum number of times a token may be used, a value of zero means unlimited

Responses

204: No Content

DELETE /auth/{approle_mount_path}/role/{role_name}/token-num-uses

Operation ID: `app-role-delete-token-num-uses`

Number of times issued tokens can be used

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: No Content

GET /auth/{approle_mount_path}/role/{role_name}/token-ttl

Operation ID: `app-role-read-token-ttl`

Duration in seconds, the lifetime of the token issued by using the SecretID that is generated against this role, before which the token needs to be renewed.

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`token_ttl`	integer	no	The initial ttl of the token to generate

POST /auth/{approle_mount_path}/role/{role_name}/token-ttl

Operation ID: `app-role-write-token-ttl`

Duration in seconds, the lifetime of the token issued by using the SecretID that is generated against this role, before which the token needs to be renewed.

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`token_ttl`	integer	no	The initial ttl of the token to generate

Responses

204: No Content

DELETE /auth/{approle_mount_path}/role/{role_name}/token-ttl

Operation ID: `app-role-delete-token-ttl`

Duration in seconds, the lifetime of the token issued by using the SecretID that is generated against this role, before which the token needs to be renewed.

Parameters

Parameter	Type	Location	Required	Description
`role_name`	string	path	yes	Name of the role. Must be less than 4096 bytes.
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: No Content

POST /auth/{approle_mount_path}/tidy/secret-id

Operation ID: `app-role-tidy-secret-id`

Trigger the clean-up of expired SecretID entries.

Parameters

Parameter	Type	Location	Required	Description
`approle_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

202: Accepted

GET /auth/{jwt_mount_path}/config

Operation ID: `jwt-read-configuration`

Read the current JWT authentication backend configuration.

Parameters

Parameter	Type	Location	Required	Description
`jwt_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /auth/{jwt_mount_path}/config

Operation ID: `jwt-configure`

Configure the JWT authentication backend.

Parameters

Parameter	Type	Location	Required	Description
`jwt_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`bound_issuer`	string	no	The value against which to match the ‘iss’ claim in a JWT. Optional.
`default_role`	string	no	The default role to use if none is provided during login. If not set, a role is required during login.
`jwks_ca_pem`	string	no	The CA certificate or chain of certificates, in PEM format, to use to validate connections to the JWKS URL. If not set, system certificates are used.
`jwks_pairs`	array	no	Set of JWKS Url and CA certificate (or chain of certificates) pairs. CA certificates must be in PEM format. Cannot be used with “jwks_url” or “jwks_ca_pem”.
`jwks_url`	string	no	JWKS URL to use to authenticate signatures. Cannot be used with “oidc_discovery_url” or “jwt_validation_pubkeys”.
`jwt_supported_algs`	array	no	A list of supported signing algorithms. Defaults to RS256.
`jwt_validation_pubkeys`	array	no	A list of PEM-encoded public keys to use to authenticate signatures locally. Cannot be used with “jwks_url” or “oidc_discovery_url”.
`namespace_in_state`	boolean	no	Pass namespace in the OIDC state parameter instead of as a separate query parameter. With this setting, the allowed redirect URL(s) in Vault and

Parameter	Type	Required	Description
			on the provider side should not contain a namespace query parameter. This means only one redirect URL entry needs to be maintained on the provider side for all vault namespaces that will be authenticating against it. Defaults to true for new configs.
`oidc_client_id`	string	no	The OAuth Client ID configured with your OIDC provider.
`oidc_client_secret`	string	no	The OAuth Client Secret configured with your OIDC provider.
`oidc_discovery_ca_pem`	string	no	The CA certificate or chain of certificates, in PEM format, to use to validate connections to the OIDC Discovery URL. If not set, system certificates are used.
`oidc_discovery_url`	string	no	OIDC Discovery URL, without any .well-known component (base path). Cannot be used with “jwks_url” or “jwt_validation_pubkeys”.
`oidc_response_mode`	string	no	The response mode to be used in the OAuth2 request. Allowed values are ‘query’ and ‘form_post’.
`oidc_response_types`	array	no	The response types to request. Allowed values are ‘code’ and ‘id_token’. Defaults to ‘code’.
`provider_config`	object	no	

Parameter	Type	Required	Description
			Provider-specific configuration. Optional.
`unsupported_critical_cert_extensions`	array	no	A list of ASN1 OIDs of certificate extensions marked Critical that are unsupported by Vault and should be ignored. This option should very rarely be needed except in specialized PKI environments.

Responses

200: OK

POST /auth/{jwt_mount_path}/login

Operation ID: `jwt-login`

Authenticates to Vault using a JWT (or OIDC) token.

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`jwt_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`distributed_claim_access_token`	string	no	An optional token used to fetch group memberships specified by the distributed claim source in the jwt. This is supported only on Azure/Entra ID
`jwt`	string	no	The signed JWT to validate.
`role`	string	no	The role to log in against.

Responses

200: OK

POST /auth/{jwt_mount_path}/oidc/auth_url

Operation ID: `jwt-oidc-request-authorization-url`

Request an authorization URL to start an OIDC login flow.

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`jwt_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`client_nonce`	string	no	Optional client-provided nonce that must match during callback, if present.
`redirect_uri`	string	no	The OAuth redirect_uri to use in the authorization URL.
`role`	string	no	The role to issue an OIDC authorization URL against.

Responses

200: OK

GET /auth/{jwt_mount_path}/oidc/callback

Operation ID: `jwt-oidc-callback`

Callback endpoint to complete an OIDC login.

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`client_nonce`	string	query	no	
`code`	string	query	no	
`state`	string	query	no	
`jwt_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /auth/{jwt_mount_path}/oidc/callback

Operation ID: `jwt-oidc-callback-form-post`

Callback endpoint to handle form_posts.

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`client_nonce`	string	query	no	
`code`	string	query	no	
`state`	string	query	no	
`jwt_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`id_token`	string	no	

Responses

200: OK

GET /auth/{jwt_mount_path}/role

Operation ID: `jwt-list-roles`

Lists all the roles registered with the backend.

Parameters

Parameter	Type	Location	Required	Description
`jwt_mount_path`	string	path	yes	Path that the backend was mounted at
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /auth/{jwt_mount_path}/role/{name}

Operation ID: `jwt-read-role`

Read an existing role.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the role.
`jwt_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /auth/{jwt_mount_path}/role/{name}

Operation ID: `jwt-write-role`

Register an role with the backend.

Creation supported: yes

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the role.
`jwt_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`allowed_redirect_uris`	array	no	Comma-separated list of allowed values for redirect_uri
`bound_audiences`	array	no	Comma-separated list of ‘aud’ claims that are valid for login; any match is sufficient
`bound_cidrs`	array	no	⚠ Deprecated. Use “token_bound_cidrs” instead. If this and “token_bound_cidrs” are both specified, only “token_bound_cidrs” will be used.
`bound_claims`	object	no	Map of claims/values which must match for login
`bound_claims_type`	string (default: string)	no	How to interpret values in the map of claims/values (which must match for login): allowed values are ‘string’ or ‘glob’
`bound_subject`	string	no	The ‘sub’ claim that is valid for login. Optional.
`claim_mappings`	object	no	Mappings of claims (key) that will be copied to a metadata field (value)
`clock_skew_leeway`	integer (default: 60000000000)	no	Duration in seconds of leeway when validating all claims to account for clock skew. Defaults to 60 (1 minute) if set to 0 and can be disabled if set to -1.
`expiration_leeway`	integer (default: 150)	no	Duration in seconds of leeway when validating expiration of a token to account for clock skew. Defaults to 150 (2.5 minutes) if set to 0 and can be disabled if set to -1.
`groups_claim`	string	no	The claim to use for the Identity group alias names
`max_age`	integer	no	

Parameter	Type	Required	Description
			Specifies the allowable elapsed time in seconds since the last time the user was actively authenticated.
`max_ttl`	integer	no	⚠ Deprecated. Use “token_max_ttl” instead. If this and “token_max_ttl” are both specified, only “token_max_ttl” will be used.
`not_before_leeway`	integer (default: 150)	no	Duration in seconds of leeway when validating not before values of a token to account for clock skew. Defaults to 150 (2.5 minutes) if set to 0 and can be disabled if set to -1.
`num_uses`	integer	no	⚠ Deprecated. Use “token_num_uses” instead. If this and “token_num_uses” are both specified, only “token_num_uses” will be used.
`oidc_scopes`	array	no	Comma-separated list of OIDC scopes
`period`	integer	no	⚠ Deprecated. Use “token_period” instead. If this and “token_period” are both specified, only “token_period” will be used.
`policies`	array	no	⚠ Deprecated. Use “token_policies” instead. If this and “token_policies” are both specified, only “token_policies” will be used.
`role_type`	string	no	Type of the role, either ‘jwt’ or ‘oidc’.
`token_bound_cidrs`	array	no	Comma separated string or JSON list of CIDR blocks. If set, specifies the blocks of IP addresses which are allowed to use the generated token.
`token_explicit_max_ttl`	integer	no	If set, tokens created via this role carry an explicit maximum TTL. During renewal, the current maximum TTL values of the role and the mount

Parameter	Type	Required	Description
			are not checked for changes, and any updates to these values will have no effect on the token being renewed.
`token_max_ttl`	integer	no	The maximum lifetime of the generated token
`token_no_default_policy`	boolean	no	If true, the ‘default’ policy will not automatically be added to generated tokens
`token_num_uses`	integer	no	The maximum number of times a token may be used, a value of zero means unlimited
`token_period`	integer	no	If set, tokens created via this role will have no max lifetime; instead, their renewal period will be fixed to this value. This takes an integer number of seconds, or a string duration (e.g. “24h”).
`token_policies`	array	no	Comma-separated list of policies
`token_ttl`	integer	no	The initial ttl of the token to generate
`token_type`	string (default: default-service)	no	The type of token to generate, service or batch
`ttl`	integer	no	⚠ Deprecated. Use “token_ttl” instead. If this and “token_ttl” are both specified, only “token_ttl” will be used.
`user_claim`	string	no	The claim to use for the Identity entity alias name
`user_claim_json_pointer`	boolean	no	If true, the user_claim value will use JSON pointer syntax for referencing claims.
`verbose_oidc_logging`	boolean	no	Log received OIDC tokens and claims when debug-level logging is active. Not recommended in production

Parameter	Type	Required	Description
			since sensitive information may be present in OIDC responses.

Responses

200: OK

DELETE /auth/{jwt_mount_path}/role/{name}

Operation ID: `jwt-delete-role`

Delete an existing role.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the role.
`jwt_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: empty body

GET /auth/{kubernetes_mount_path}/config

Operation ID: `kubernetes-read-auth-configuration`

Configures the JWT Public Key and Kubernetes API information.

Parameters

Parameter	Type	Location	Required	Description
`kubernetes_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /auth/{kubernetes_mount_path}/config

Operation ID: `kubernetes-configure-auth`

Configures the JWT Public Key and Kubernetes API information.

Parameters

Parameter	Type	Location	Required	Description
`kubernetes_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`disable_iss_validation`	boolean (default: True)	no	⚠ Deprecated. Disable JWT issuer validation (Deprecated, will be removed in a future release)
`disable_local_ca_jwt`	boolean (default: False)	no	Disable defaulting to the local CA cert and service account JWT when running in a Kubernetes pod
`issuer`	string	no	⚠ Deprecated. Optional JWT issuer. If no issuer is specified, then this plugin will use kubernetes.io/serviceaccount as the default issuer. (Deprecated, will be removed in a future release)
`kubernetes_ca_cert`	string	no	Optional PEM encoded CA cert for use by the TLS client used to talk with the API. If it is not set and disable_local_ca_jwt is true, the system’s trusted CA certificate pool will be used.
`kubernetes_host`	string	no	Host must be a host string, a host:port pair, or a URL to the base of the Kubernetes API server.
`pem_keys`	array	no	Optional list of PEM-formated public keys or certificates used to verify the signatures of kubernetes service account JWTs. If a certificate is given, its public key will be extracted. Not every installation of Kubernetes exposes these keys.
`token_reviewer_jwt`	string	no	A service account JWT (or other token) used as a bearer token to access the TokenReview API to

Parameter	Type	Required	Description
			validate other JWTs during login. If not set the JWT used for login will be used to access the API.
`use_annotations_as_alias_metadata`	boolean (default: False)	no	Use annotations from the client token's associated service account as alias metadata for the Vault entity. Only annotations with the prefix "vault.hashicorp.com/alias-metadata-" will be used. Note that Vault will need permission to read service accounts from the Kubernetes API.

Responses

200: OK

POST /auth/{kubernetes_mount_path}/login

Operation ID: `kubernetes-login`

Authenticates Kubernetes service accounts with Vault.

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`kubernetes_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`jwt`	string	no	A signed JWT for authenticating a service account. This field is required.
`role`	string	no	Name of the role against which the login is being attempted. This field is required

Responses

200: OK

GET /auth/{kubernetes_mount_path}/role

Operation ID: `kubernetes-list-auth-roles`

Lists all the roles registered with the backend.

Parameters

Parameter	Type	Location	Required	Description
`kubernetes_mount_path`	string	path	yes	Path that the backend was mounted at
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /auth/{kubernetes_mount_path}/role/{name}

Operation ID: `kubernetes-read-auth-role`

Register an role with the backend.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the role.
`kubernetes_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /auth/{kubernetes_mount_path}/role/{name}

Operation ID: `kubernetes-write-auth-role`

Register an role with the backend.

Creation supported: yes

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the role.
`kubernetes_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`alias_name_source`	string (default: serviceaccount_uid)	no	Source to use when deri name. valid choices: “serviceaccount_uid” : < 474b11b5-0f20-4f9d-8ca (most secure choice) “serviceaccount_name” vault-agent default: “serv
`audience`	string	no	Optional Audience claim jwt.
`bound_cidrs`	array	no	⚠ Deprecated. Use “toke instead. If this and “toke are both specified, only “token_bound_cidrs” will
`bound_service_account_names`	array	no	List of service account n access this role. If set to allowed.
`bound_service_account_namespace_selector`	string	no	A label selector for Kube namespaces which are a this role. Accepts either object. If set with bound_service_account the conditions are ORed
`bound_service_account_namespaces`	array	no	List of namespaces allow this role. If set to “*” all n allowed.
`max_ttl`	integer	no	⚠ Deprecated. Use “toke instead. If this and “toke both specified, only “toke be used.
`num_uses`	integer	no	⚠ Deprecated. Use “toke instead. If this and “toke both specified, only “toke will be used.
`period`	integer	no	⚠ Deprecated. Use “toke instead. If this and “toke

Parameter	Type	Required	Description
			both specified, only “toke used.
`policies`	array	no	⚠ Deprecated. Use “toke instead. If this and “token both specified, only “toke be used.
`token_bound_cidrs`	array	no	Comma separated string CIDR blocks. If set, spec of IP addresses which ar the generated token.
`token_explicit_max_ttl`	integer	no	If set, tokens created via an explicit maximum TTL renewal, the current max values of the role and th checked for changes, ar these values will have no token being renewed.
`token_max_ttl`	integer	no	The maximum lifetime of token
`token_no_default_policy`	boolean	no	If true, the ‘default’ policy automatically be added t tokens
`token_num_uses`	integer	no	The maximum number o may be used, a value of unlimited
`token_period`	integer	no	If set, tokens created via have no max lifetime; ins renewal period will be fix This takes an integer nu or a string duration (e.g.
`token_policies`	array	no	Comma-separated list of
`token_ttl`	integer	no	The initial ttl of the token
`token_type`	string (default: default-service)	no	The type of token to gen batch

Parameter	Type	Required	Description
`ttl`	integer	no	⚠ Deprecated. Use “tok this and “token_ttl” are b only “token_ttl” will be us

Responses

200: OK

DELETE /auth/{kubernetes_mount_path}/role/{name}

Operation ID: `kubernetes-delete-auth-role`

Register an role with the backend.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the role.
`kubernetes_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: empty body

GET /auth/{ldap_mount_path}/config

Operation ID: `ldap-read-auth-configuration`

Configure the LDAP server to connect to, along with its options.

Parameters

Parameter	Type	Location	Required	Description
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /auth/{ldap_mount_path}/config

Operation ID: `ldap-configure-auth`

Configure the LDAP server to connect to, along with its options.

Parameters

Parameter	Type	Location	Required	Description
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`anonymous_group_search`	boolean (default: False)	no	Use anonymous binds whe LDAP group searches (if tru credentials will still be used connection test).
`binddn`	string	no	LDAP DN for searching for (optional)
`bindpass`	string	no	LDAP password for searchi user DN (optional)
`case_sensitive_names`	boolean	no	If true, case sensitivity will b when comparing username groups for matching policies
`certificate`	string	no	CA certificate to use when v LDAP server certificate, mu PEM encoded (optional)
`client_tls_cert`	string	no	Client certificate to provide server, must be x509 PEM (optional)
`client_tls_key`	string	no	Client certificate key to prov LDAP server, must be x509 encoded (optional)
`connection_timeout`	integer (default: 30s)	no	Timeout, in seconds, when to connect to the LDAP serv trying the next URL in the c
`deny_null_bind`	boolean (default: True)	no	⚠ Deprecated. Denies an unauthenticated LDAP bind the user's password is emp to true
`dereference_aliases`	string (never, finding, searching, always) (default: never)	no	When aliases should be de on search operations. Acce are 'never', 'finding', 'search 'always'. Defaults to 'never'
`discoverdn`	boolean	no	Use anonymous bind to dis bind DN of a user (optional)

Parameter	Type	Required	Description
`enable_samaccountname_login`	boolean (default: False)	no	If true, matching sAMAccou attribute values will be allow when upndomain is defined
`groupattr`	string (default: cn)	no	LDAP attribute to follow on returned by in order to enur group membership. Examp “memberOf”, etc. Default: c
`groupdn`	string	no	LDAP search base to use fo membership search (eg: ou=Groups,dc=example,dc
`groupfilter`	string (default: (| (memberUid={{.Username}}) (member={{.UserDN}}) (uniqueMember={{.UserDN}})))	no	Go template for querying gr membership of user (option template can access the fol context variables: UserDN, Example: (&(objectClass=g (member: 1.2.840.113556.1.4.1941:={ Default: (|(memberUid={{.U (member={{.UserDN}}) (uniqueMember={{.UserDN
`insecure_tls`	boolean	no	Skip LDAP server SSL Cert verification - VERY insecure
`max_page_size`	integer (default: 0)	no	If set to a value greater than LDAP backend will use the server’s paged search cont request pages of up to the g This can be used to avoid h LDAP server’s maximum re limit. Otherwise, the LDAP b not use the paged search c
`request_timeout`	integer (default: 90s)	no	Timeout, in seconds, for the when making requests agai server before returning bac
`starttls`	boolean	no	Issue a StartTLS command establishing unencrypted co (optional)

Parameter	Type	Required	Description
`tls_max_version`	string (tls10, tls11, tls12, tls13) (default: tls12)	no	Maximum TLS version to us values are ’tls10’, ’tls11’, ’tls ’tls13’. Defaults to ’tls12’
`tls_min_version`	string (tls10, tls11, tls12, tls13) (default: tls12)	no	Minimum TLS version to us values are ’tls10’, ’tls11’, ’tls ’tls13’. Defaults to ’tls12’
`token_bound_cidrs`	array	no	Comma separated string or CIDR blocks. If set, specifie of IP addresses which are a use the generated token.
`token_explicit_max_ttl`	integer	no	If set, tokens created via thi an explicit maximum TTL. D renewal, the current maxim values of the role and the m checked for changes, and a to these values will have no the token being renewed.
`token_max_ttl`	integer	no	The maximum lifetime of th token
`token_no_default_policy`	boolean	no	If true, the ‘default’ policy w automatically be added to g tokens
`token_num_uses`	integer	no	The maximum number of tim may be used, a value of zer unlimited
`token_period`	integer	no	If set, tokens created via thi have no max lifetime; instea renewal period will be fixed value. This takes an integer seconds, or a string duratio “24h”).
`token_policies`	array	no	Comma-separated list of po will apply to all tokens gene auth method, in addition to configured for specific users

Parameter	Type	Required	Description
`token_ttl`	integer	no	The initial ttl of the token to
`token_type`	string (default: default-service)	no	The type of token to genera or batch
`upndomain`	string	no	Enables userPrincipalDoma [username]@UPNDomain (
`url`	string (default: ldap:// 127.0.0.1)	no	LDAP URL to connect to (d 127.0.0.1). Multiple URLs c specified by concatenating commas; they will be tried i
`use_pre111_group_cn_behavior`	boolean	no	In Vault 1.1.1 a fix for handl CN values of different case unfortunately introduced a r that could cause previously groups to not be found due in the resulting name. If set pre-1.1.1 behavior for matc CNs will be used. This is or some upgrade scenarios fo compatibility. It is enabled b the config is upgraded but c default on new configuratio
`use_token_groups`	boolean (default: False)	no	If true, use the Active Direct tokenGroups constructed a the user to find the group m This will find all security gro including nested ones.
`userattr`	string (default: cn)	no	Attribute used for users (de
`userdn`	string	no	LDAP domain to use for us ou=People,dc=example,dc=
`userfilter`	string (default: ({{.UserAttr}} ={{.Username}}))	no	Go template for LDAP user (optional) The template can following context variables: Username Default: ({{.User ={{.Username}})

Parameter	Type	Required	Description
`username_as_alias`	boolean (default: False)	no	If true, sets the alias name to username

Responses

200: OK

GET /auth/{ldap_mount_path}/groups

Operation ID: `ldap-list-groups`

Manage additional groups for users allowed to authenticate.

Parameters

Parameter	Type	Location	Required	Description
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /auth/{ldap_mount_path}/groups/{name}

Operation ID: `ldap-read-group`

Manage additional groups for users allowed to authenticate.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the LDAP group.
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /auth/{ldap_mount_path}/groups/{name}

Operation ID: `ldap-write-group`

Manage additional groups for users allowed to authenticate.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the LDAP group.
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`policies`	array	no	Comma-separated list of policies associated to the group.

Responses

200: OK

DELETE /auth/{ldap_mount_path}/groups/{name}

Operation ID: `ldap-delete-group`

Manage additional groups for users allowed to authenticate.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the LDAP group.
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: empty body

POST /auth/{ldap_mount_path}/login/{username}

Operation ID: `ldap-login`

Log in with a username and password.

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`username`	string	path	yes	DN (distinguished name) to be used for login.
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`password`	string	no	Password for this user.

Responses

200: OK

GET /auth/{ldap_mount_path}/users

Operation ID: `ldap-list-users`

Manage users allowed to authenticate.

Parameters

Parameter	Type	Location	Required	Description
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /auth/{ldap_mount_path}/users/{name}

Operation ID: `ldap-read-user`

Manage users allowed to authenticate.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the LDAP user.
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /auth/{ldap_mount_path}/users/{name}

Operation ID: `ldap-write-user`

Manage users allowed to authenticate.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the LDAP user.
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`groups`	array	no	Comma-separated list of additional groups associated with the user.
`policies`	array	no	Comma-separated list of policies associated with the user.

Responses

200: OK

DELETE /auth/{ldap_mount_path}/users/{name}

Operation ID: `ldap-delete-user`

Manage users allowed to authenticate.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the LDAP user.
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: empty body

POST /auth/{saml_mount_path}/callback

Operation ID: `saml-write-callback`

Provides the Assertion Consumer Service to handle the Identity Provider binding.

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`saml_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`RelayState`	string	yes	The SAML RelayState that has round tripped through the Identity Provider.
`SAMLResponse`	string	yes	The SAML response from the Identity Provider.

Responses

200: OK

GET /auth/{saml_mount_path}/config

Operation ID: `saml-read-config`

Read the configuration of the auth method.

Parameters

Parameter	Type	Location	Required	Description
`saml_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /auth/{saml_mount_path}/config

Operation ID: `saml-write-config`

Update the configuration of the auth method.

Creation supported: yes

Parameters

Parameter	Type	Location	Required	Description
`saml_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`acs_urls`	array	yes	The Assertion Consumer Service URLs to which the responses from the Identity Provider will be sent. Must be well-formatted URLs. A warning will be provided if any of the given URLs are not TLS protected.
`default_role`	string	no	The role to use if no role is provided during login. If not set, a role is required during login.
`entity_id`	string	yes	The entity ID of this authentication method as a SAML Service Provider
`idp_cert`	string	no	The PEM-encoded certificate of the Identity Provider used to verify response and assertion signatures. Mutually exclusive with ‘idp_metadata_url’.
`idp_entity_id`	string	no	The entity ID of the Identity Provider. Mutually exclusive with ‘idp_metadata_url’.
`idp_metadata_url`	string	no	The metadata URL of the Identity Provider. Mutually exclusive with ‘idp_sso_url’, ‘idp_issuer’ and ‘idp_cert’. Must be a well-formatted URL.
`idp_sso_url`	string	no	The SSO URL of the Identity Provider. Mutually exclusive with ‘idp_metadata_url’. Must be a well-formatted URL.
`validate_assertion_signature`	boolean	no	Enables validation of signature for at least assertion in the SAML response. If IDP allows signing both response and assertion, then recommendation is to opt for validating signatures of both by enabling individual options
`validate_response_signature`	boolean	no	Enables validation of signature for at least response in the SAML response.

Parameter	Type	Required	Description
			If IDP allows signing both response and assertion, then recommendation is to opt for validating signatures of both by enabling individual options
`verbose_logging`	boolean	no	Log additional information during the SAML exchange. The user data will be logged when debug-level logging is active and the full SAML response will be logged when trace-level logging is active

Responses

200: OK

GET /auth/{saml_mount_path}/role

Operation ID: `saml-list-role`

List all roles.

Parameters

Parameter	Type	Location	Required	Description
`saml_mount_path`	string	path	yes	Path that the backend was mounted at
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /auth/{saml_mount_path}/role/{name}

Operation ID: `saml-read-role-name`

Read a role’s configuration.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	The name of the role.
`saml_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /auth/{saml_mount_path}/role/{name}

Operation ID: `saml-write-role-name`

Update a role's configuration.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	The name of the role.
`saml_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`alias_metadata`	object	no	The metadata to be tied to generated entity alias. This should be a list or map containing the metadata in key value pairs
`bound_attributes`	object	no	Mapping of attribute names to values to assert exist in the SAML Response's Assertion.
`bound_attributes_type`	string (string, glob)	no	The type of matching assertion to perform on bound_attributes key-value pairs. If 'string', requires a direct string match in values. If 'glob', allows for wildcard matching using the '*' character in values.
`bound_subjects`	array	no	The subject to assert is in the SAML Response. The subject in theSAML Response needs to match one of the values configured.
`bound_subjects_type`	string (string, glob)	no	The type of matching assertion to perform on bound_subject. If 'string', requires a direct string match. If 'glob', allows for wildcardmatching using the '*' character.
`groups_attribute`	string	no	The attribute to use for Vault Identity group alias names.
`token_bound_cidrs`	array	no	Comma separated string or JSON list of CIDR blocks. If set, specifies the blocks of IP addresses which are allowed to use the generated token.
`token_explicit_max_ttl`	integer	no	If set, tokens created via this role carry an explicit maximum TTL. During renewal, the current maximum TTL values of the role and the mount are not checked for changes, and any updates to these values will have no effect on the token being renewed.
`token_max_ttl`	integer	no	The maximum lifetime of the generated token

Parameter	Type	Required	Description
`token_no_default_policy`	boolean	no	If true, the ‘default’ policy will not automatically be added to generated tokens
`token_num_uses`	integer	no	The maximum number of times a token may be used, a value of zero means unlimited
`token_period`	integer	no	If set, tokens created via this role will have no max lifetime; instead, their renewal period will be fixed to this value. This takes an integer number of seconds, or a string duration (e.g. “24h”).
`token_policies`	array	no	Comma-separated list of policies
`token_ttl`	integer	no	The initial ttl of the token to generate
`token_type`	string (default: default-service)	no	The type of token to generate, service or batch

Responses

200: OK

DELETE /auth/{saml_mount_path}/role/{name}

Operation ID: `saml-delete-role-name`

Delete a role.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	The name of the role.
`saml_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: empty body

POST /auth/{saml_mount_path}/sso_service_url

Operation ID: `saml-write-sso_service_url`

Obtain an SSO Service URL to start a SAML authentication flow.

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`saml_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`acs_url`	string	no	The Assertion Consumer Service URL to use for this auth request. It must be one of the allowed URLs in the config.
`client_challenge`	string	yes	The client challenge. Must be the output of a base64-encoded, sha256 digest of the 'client_verifier' eventually provided to the token API.
`client_type`	string (default: cli)	no	The type of the requesting client. The response from the Assertion Consumer Service callback API will differ based on the provided type.
`role`	string	no	The role to issue an SSO Service URL for.

Responses

200: OK

POST /auth/{saml_mount_path}/token

Operation ID: `saml-write-token`

Obtain a Vault token to complete the authentication flow.

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`saml_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`client_verifier`	string	yes	The value which produced the ‘client_challenge’ provided to the SSO Service URL API at the start of the authentication flow. Its base64-encoded, sha256 digest must match the ‘client_challenge’ value.
`token_poll_id`	string	yes	The ’token_poll_id’ value returned from the SSO Service URL API at the start of the authentication flow.

Responses

200: OK

POST /auth/{userpass_mount_path}/login/{username}

Operation ID: `userpass-login`

Log in with a username and password.

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`username`	string	path	yes	Username of the user.
`userpass_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`password`	string	no	Password for this user.

Responses

200: OK

POST /auth/{userpass_mount_path}/password_policy/{policy_name}

Operation ID: `userpass-update-policies_password`

Parameters

Parameter	Type	Location	Required	Description
`policy_name`	string	path	yes	Policy password name for userpass.
`userpass_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

GET /auth/{userpass_mount_path}/users

Operation ID: `userpass-list-users`

Manage users allowed to authenticate.

Parameters

Parameter	Type	Location	Required	Description
`userpass_mount_path`	string	path	yes	Path that the backend was mounted at
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /auth/{userpass_mount_path}/users/{username}

Operation ID: `userpass-read-user`

Manage users allowed to authenticate.

Parameters

Parameter	Type	Location	Required	Description
`username`	string	path	yes	Username for this user.
`userpass_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /auth/{userpass_mount_path}/users/{username}

Operation ID: `userpass-write-user`

Manage users allowed to authenticate.

Creation supported: yes

Parameters

Parameter	Type	Location	Required	Description
`username`	string	path	yes	Username for this user.
`userpass_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`bound_cidrs`	array	no	⚠ Deprecated. Use “token_bound_cidrs” instead. If this and “token_bound_cidrs” are both specified, only “token_bound_cidrs” will be used.
`max_ttl`	integer	no	⚠ Deprecated. Use “token_max_ttl” instead. If this and “token_max_ttl” are both specified, only “token_max_ttl” will be used.
`password`	string	no	Password for this user.
`policies`	array	no	⚠ Deprecated. Use “token_policies” instead. If this and “token_policies” are both specified, only “token_policies” will be used.
`token_bound_cidrs`	array	no	Comma separated string or JSON list of CIDR blocks. If set, specifies the blocks of IP addresses which are allowed to use the generated token.
`token_explicit_max_ttl`	integer	no	If set, tokens created via this role carry an explicit maximum TTL. During renewal, the current maximum TTL values of the role and the mount are not checked for changes, and any updates to these values will have no effect on the token being renewed.
`token_max_ttl`	integer	no	The maximum lifetime of the generated token
`token_no_default_policy`	boolean	no	If true, the ‘default’ policy will not automatically be added to generated tokens
`token_num_uses`	integer	no	The maximum number of times a token may be used, a value of zero means unlimited
`token_period`	integer	no	If set, tokens created via this role will have no max lifetime; instead, their renewal period will be fixed to this value. This takes

Parameter	Type	Required	Description
			an integer number of seconds, or a string duration (e.g. “24h”).
`token_policies`	array	no	Comma-separated list of policies
`token_ttl`	integer	no	The initial ttl of the token to generate
`token_type`	string (default: default-service)	no	The type of token to generate, service or batch
`ttl`	integer	no	⚠ Deprecated. Use “token_ttl” instead. If this and “token_ttl” are both specified, only “token_ttl” will be used.

Responses

200: OK

DELETE /auth/{userpass_mount_path}/users/{username}

Operation ID: `userpass-delete-user`

Manage users allowed to authenticate.

Parameters

Parameter	Type	Location	Required	Description
`username`	string	path	yes	Username for this user.
`userpass_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: empty body

POST /auth/{userpass_mount_path}/users/{username}/password

Operation ID: `userpass-reset-password`

Reset user’s password.

Parameters

Parameter	Type	Location	Required	Description
`username`	string	path	yes	Username for this user.
`userpass_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`password`	string	no	Password for this user.

Responses

200: OK

POST /auth/{userpass_mount_path}/users/{username}/policies

Operation ID: `userpass-update-policies`

Update the policies associated with the username.

Parameters

Parameter	Type	Location	Required	Description
`username`	string	path	yes	Username for this user.
`userpass_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`policies`	array	no	⚠ Deprecated. Use “token_policies” instead. If this and “token_policies” are both specified, only “token_policies” will be used.
`token_policies`	array	no	Comma-separated list of policies

Responses

200: OK

GET /auth/{webauthn_mount_path}/config

Operation ID: `webauthn-read-config`

Read WebAuthn configuration

Parameters

Parameter	Type	Location	Required	Description
`webauthn_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /auth/{webauthn_mount_path}/config

Operation ID: `webauthn-write-config`

Configure WebAuthn backend

Parameters

Parameter	Type	Location	Required	Description
`webauthn_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`auto_registration`	boolean	no	If true (default), new users can self-register. If false, only pre-created users (via user/ path) can register.
`rp_display_name`	string	no	Human-readable name for the Relying Party.
`rp_id`	string	no	Relying Party ID (e.g. localhost or your domain). Must match the origin's host.
`rp_origins`	array	no	Allowed origins for WebAuthn (e.g. https://vault.example.com, http://localhost:8200).

Responses

200: OK

POST /auth/{webauthn_mount_path}/login/begin

Operation ID: `webauthn-write-login-begin`

Start WebAuthn login

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`webauthn_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`username`	string	no	Username to authenticate (omit for discoverable/passkey flow: browser will show passkey picker)

Responses

200: OK

POST /auth/{webauthn_mount_path}/login/finish

Operation ID: `webauthn-write-login-finish`

Finish WebAuthn login

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`webauthn_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`credential`	object	no	Assertion response from the authenticator (PublicKeyCredential with response)
`username`	string	no	Username (omit if using discoverable flow; user is identified from assertion userHandle)

Responses

200: OK

POST /auth/{webauthn_mount_path}/register/begin

Operation ID: `webauthn-write-register-begin`

Start WebAuthn registration

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`webauthn_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`username`	string	no	Username to register for WebAuthn

Responses

200: OK

POST /auth/{webauthn_mount_path}/register/finish

Operation ID: `webauthn-write-register-finish`

Finish WebAuthn registration

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`webauthn_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`credential`	object	no	Credential creation response from the authenticator (PublicKeyCredential with response)
`username`	string	no	Username being registered

Responses

200: OK

GET /auth/{webauthn_mount_path}/user

Operation ID: `webauthn-list-user`

List registered users

Parameters

Parameter	Type	Location	Required	Description
`webauthn_mount_path`	string	path	yes	Path that the backend was mounted at
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /auth/{webauthn_mount_path}/user/{name}

Operation ID: `webauthn-read-user-name`

Read a registered user

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Username
`webauthn_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /auth/{webauthn_mount_path}/user/{name}

Operation ID: `webauthn-write-user-name`

Update a user

Creation supported: yes

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Username
`webauthn_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`display_name`	string	no	Display name for the user (defaults to username)
`token_bound_cidrs`	array	no	Comma separated string or JSON list of CIDR blocks. If set, specifies the blocks of IP addresses which are allowed to use the generated token.
`token_max_ttl`	integer	no	The maximum lifetime of the generated token
`token_no_default_policy`	boolean	no	If true, the ‘default’ policy will not automatically be added to generated tokens
`token_period`	integer	no	If set, tokens created via this role will have no max lifetime; instead, their renewal period will be fixed to this value. This takes an integer number of seconds, or a string duration (e.g. “24h”).
`token_policies`	array	no	Comma-separated list of policies
`token_ttl`	integer	no	The initial ttl of the token to generate

Responses

200: OK

DELETE /auth/{webauthn_mount_path}/user/{name}

Operation ID: `webauthn-delete-user-name`

Delete a registered user

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Username
`webauthn_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: empty body

DELETE /auth/{webauthn_mount_path}/user/{name}/credential/{credential_id}

Operation ID: `webauthn-delete-user-name-credential-credential_id`

Remove a credential from a user

Parameters

Parameter	Type	Location	Required	Description
`credential_id`	string	path	yes	Credential ID (base64url encoded)
`name`	string	path	yes	Username
`webauthn_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: empty body

POST /auth/{webauthn_mount_path}/user/{name}/policies

Operation ID: `webauthn-write-user-name-policies`

Update user token policies

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Username
`webauthn_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`token_policies`	array	no	Comma-separated list of policies for the generated token

Responses

200: OK

secrets

secrets

GET /cubbyhole/{path}

Operation ID: `cubbyhole-read`

Retrieve the secret at the specified location.

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	Specifies the path of the secret.
`list`	string	query	no	Return a list if `true`

Responses

200: OK

POST /cubbyhole/{path}

Operation ID: `cubbyhole-write`

Store a secret at the specified location.

Creation supported: yes

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	Specifies the path of the secret.

Responses

200: OK

DELETE /cubbyhole/{path}

Operation ID: `cubbyhole-delete`

Deletes the secret at the specified location.

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	Specifies the path of the secret.

Responses

204: empty body

GET /{database_mount_path}/config

Operation ID: `database-list-connections`

Configure connection details to a database plugin.

Parameters

Parameter	Type	Location	Required	Description
`database_mount_path`	string	path	yes	Path that the backend was mounted at
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /{database_mount_path}/config/{name}

Operation ID: `database-read-connection-configuration`

Configure connection details to a database plugin.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of this database connection
`database_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /{database_mount_path}/config/{name}

Operation ID: `database-configure-connection`

Configure connection details to a database plugin.

Creation supported: yes

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of this database connection
`database_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`allowed_roles`	array	no	Comma separated string or array of the role names allowed to get creds from this database connection. If empty no roles are allowed. If “*” all roles are allowed.
`password_policy`	string	no	Password policy to use when generating passwords.
`plugin_name`	string	no	The name of a builtin or previously registered plugin known to vault. This endpoint will create an instance of that plugin type.
`plugin_version`	string	no	The version of the plugin to use.
`root_rotation_statements`	array	no	Specifies the database statements to be executed to rotate the root user’s credentials. See the plugin’s API page for more information on support and formatting for this parameter.
`verify_connection`	boolean (default: True)	no	If true, the connection details are verified by actually connecting to the database. Defaults to true.

Responses

200: OK

DELETE /{database_mount_path}/config/{name}

Operation ID: `database-delete-connection-configuration`

Configure connection details to a database plugin.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of this database connection
`database_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: empty body

GET /{database_mount_path}/creds/{name}

Operation ID: `database-generate-credentials`

Request database credentials for a certain role.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the role.
`database_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /{database_mount_path}/reset/{name}

Operation ID: `database-reset-connection`

Resets a database plugin.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of this database connection
`database_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

GET /{database_mount_path}/roles

Operation ID: `database-list-roles`

Manage the roles that can be created with this backend.

Parameters

Parameter	Type	Location	Required	Description
`database_mount_path`	string	path	yes	Path that the backend was mounted at
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /{database_mount_path}/roles/{name}

Operation ID: `database-read-role`

Manage the roles that can be created with this backend.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the role.
`database_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /{database_mount_path}/roles/{name}

Operation ID: `database-write-role`

Manage the roles that can be created with this backend.

Creation supported: yes

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the role.
`database_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`creation_statements`	array	no	Specifies the database statements executed to create and configure a user. See the plugin's API page for more information on support and formatting for this parameter.
`credential_config`	object	no	The configuration for the given credential_type.
`credential_type`	string (default: password)	no	The type of credential to manage. Options include: 'password', 'rsa_private_key'. Defaults to 'password'.
`db_name`	string	no	Name of the database this role acts on.
`default_ttl`	integer	no	Default ttl for role.
`max_ttl`	integer	no	Maximum time a credential is valid for
`renew_statements`	array	no	Specifies the database statements to be executed to renew a user. Not every plugin type will support this functionality. See the plugin's API page for more information on support and formatting for this parameter.
`revocation_statements`	array	no	Specifies the database statements to be executed to revoke a user. See the plugin's API page for more information on support and formatting for this parameter.
`rollback_statements`	array	no	Specifies the database statements to be executed rollback a create operation in the event of an error. Not every plugin type will support this functionality. See the plugin's API page for more information on support and formatting for this parameter.

Responses

200: OK

DELETE /{database_mount_path}/roles/{name}

Operation ID: `database-delete-role`

Manage the roles that can be created with this backend.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the role.
`database_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: empty body

POST /{database_mount_path}/rotate-role/{name}

Operation ID: `database-rotate-static-role-credentials`

Request to rotate the credentials for a static user account.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the static role
`database_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /{database_mount_path}/rotate-root/{name}

Operation ID: `database-rotate-root-credentials`

Request to rotate the root credentials for a certain database connection.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of this database connection
`database_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

GET /{database_mount_path}/static-creds/{name}

Operation ID: `database-read-static-role-credentials`

Request database credentials for a certain static role. These credentials are rotated periodically.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the static role.
`database_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

GET /{database_mount_path}/static-roles

Operation ID: `database-list-static-roles`

Manage the static roles that can be created with this backend.

Parameters

Parameter	Type	Location	Required	Description
`database_mount_path`	string	path	yes	Path that the backend was mounted at
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /{database_mount_path}/static-roles/{name}

Operation ID: `database-read-static-role`

Manage the static roles that can be created with this backend.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the role.
`database_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /{database_mount_path}/static-roles/{name}

Operation ID: `database-write-static-role`

Manage the static roles that can be created with this backend.

Creation supported: yes

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the role.
`database_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`credential_config`	object	no	The configuration for the given credential_type.
`credential_type`	string (default: password)	no	The type of credential to manage. Options include: ‘password’, ‘rsa_private_key’. Defaults to ‘password’.
`db_name`	string	no	Name of the database this role acts on.
`rotation_period`	integer	no	Period for automatic credential rotation of the given username. Not valid unless used with “username”.
`rotation_statements`	array	no	Specifies the database statements to be executed to rotate the accounts credentials. Not every plugin type will support this functionality. See the plugin’s API page for more information on support and formatting for this parameter.
`username`	string	no	Name of the static user account for Vault to manage. Requires “rotation_period” to be specified

Responses

200: OK

DELETE /{database_mount_path}/static-roles/{name}

Operation ID: `database-delete-static-role`

Manage the static roles that can be created with this backend.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the role.
`database_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: empty body

GET /{kubernetes_mount_path}/check

Operation ID: `kubernetes-check-configuration`

Checks the Kubernetes configuration is valid.

Parameters

Parameter	Type	Location	Required	Description
`kubernetes_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

GET /{kubernetes_mount_path}/config

Operation ID: `kubernetes-read-configuration`

Configure the Kubernetes secret engine plugin.

Parameters

Parameter	Type	Location	Required	Description
`kubernetes_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /{kubernetes_mount_path}/config

Operation ID: `kubernetes-configure`

Configure the Kubernetes secret engine plugin.

Parameters

Parameter	Type	Location	Required	Description
`kubernetes_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`disable_local_ca_jwt`	boolean (default: False)	no	Disable defaulting to the local CA certificate and service acc when running in a Kubernetes pod.
`kubernetes_ca_cert`	string	no	PEM encoded CA certificate to use to verify the Kubernetes server certificate. Defaults to the local pod's CA if found.
`kubernetes_host`	string	no	Kubernetes API URL to connect to. Defaults to https:// $KUBERNETES_SERVICE_HOST:KUBERNETES_SERVIC if those environment variables are set.
`service_account_jwt`	string	no	The JSON web token of the service account used by the se engine to manage Kubernetes credentials. Defaults to the lo JWT if found.

Responses

200: OK

DELETE /{kubernetes_mount_path}/config

Operation ID: `kubernetes-delete-configuration`

Configure the Kubernetes secret engine plugin.

Parameters

Parameter	Type	Location	Required	Description
`kubernetes_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: empty body

POST /{kubernetes_mount_path}/creds/{name}

Operation ID: `kubernetes-generate-credentials`

Request Kubernetes service account credentials for a given Vault role.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the Vault role
`kubernetes_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`audiences`	array	no	The intended audiences of the generated credentials
`cluster_role_binding`	boolean	no	If true, generate a ClusterRoleBinding to grant permissions across the whole cluster instead of within a namespace. Requires the Vault role to have kubernetes_role_type set to ClusterRole.
`kubernetes_namespace`	string	yes	The name of the Kubernetes namespace in which to generate the credentials
`ttl`	integer	no	The TTL of the generated credentials

Responses

200: OK

GET /{kubernetes_mount_path}/roles

Operation ID: `kubernetes-list-roles`

List the existing roles in this secrets engine.

Parameters

Parameter	Type	Location	Required	Description
`kubernetes_mount_path`	string	path	yes	Path that the backend was mounted at
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /{kubernetes_mount_path}/roles/{name}

Operation ID: `kubernetes-read-role`

Manage the roles that can be created with this secrets engine.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the role
`kubernetes_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /{kubernetes_mount_path}/roles/{name}

Operation ID: `kubernetes-write-role`

Manage the roles that can be created with this secrets engine.

Creation supported: yes

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the role
`kubernetes_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`allowed_kubernetes_namespace_selector`	string	no	A label selector for Kubernetes namespaces in which credentials can be generated. Accepts either a JSON or YAML object. If set with allowed_kubernetes_namespaces, the conditions are conjuncted.
`allowed_kubernetes_namespaces`	array	no	A list of the Kubernetes namespaces in which credentials can be generated. If set to “*” all namespaces are allowed.
`extra_annotations`	object	no	Additional annotations to apply to all generated Kubernetes objects.
`extra_labels`	object	no	Additional labels to apply to all generated Kubernetes objects.
`generated_role_rules`	string	no	The Role or ClusterRole rules to use when generating a role. Accepts either a JSON or YAML object. If set, the entire chain of Kubernetes objects will be generated.
`kubernetes_role_name`	string	no	The pre-existing Role or ClusterRole to bind a generated service account to. If set, Kubernetes token, service account, and role binding objects will be created.
`kubernetes_role_type`	string (default: Role)	no	Specifies whether the Kubernetes role is a Role or ClusterRole.
`name_template`	string	no	The name template to use when generating service accounts, roles and role bindings. If unset, a default template is used.
`service_account_name`	string	no	The pre-existing service account to generate tokens for. Mutually

Parameter	Type	Required	Description
			exclusive with all role parameters. If set, only a Kubernetes service account token will be created.
`token_default_audiences`	array	no	The default audiences for generated Kubernetes service account tokens. If not set or set to “”, will use k8s cluster default.
`token_default_ttl`	integer	no	The default ttl for generated Kubernetes service account tokens. If not set or set to 0, will use system default.
`token_max_ttl`	integer	no	The maximum ttl for generated Kubernetes service account tokens. If not set or set to 0, will use system default.

Responses

200: OK

DELETE /{kubernetes_mount_path}/roles/{name}

Operation ID: `kubernetes-delete-role`

Manage the roles that can be created with this secrets engine.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the role
`kubernetes_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: empty body

GET /{kv_v1_mount_path}/{path}

Operation ID: `kv-v1-read`

Pass-through secret storage to the storage backend, allowing you to read/write arbitrary data into secret storage.

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	Location of the secret.
`kv_v1_mount_path`	string	path	yes	Path that the backend was mounted at
`list`	string	query	no	Return a list if `true`

Responses

200: OK

POST /{kv_v1_mount_path}/{path}

Operation ID: `kv-v1-write`

Pass-through secret storage to the storage backend, allowing you to read/write arbitrary data into secret storage.

Creation supported: yes

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	Location of the secret.
`kv_v1_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: No Content

DELETE /{kv_v1_mount_path}/{path}

Operation ID: `kv-v1-delete`

Pass-through secret storage to the storage backend, allowing you to read/write arbitrary data into secret storage.

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	Location of the secret.
`kv_v1_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: No Content

GET /{kv_v2_mount_path}/config

Operation ID: `kv-v2-read-configuration`

Read the backend level settings.

Parameters

Parameter	Type	Location	Required	Description
`kv_v2_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`cas_required`	boolean	no	If true, the backend will require the cas parameter to be set for each write
`delete_version_after`	integer	no	The length of time before a version is deleted.
`max_versions`	integer	no	The number of versions to keep for each key.

POST /{kv_v2_mount_path}/config

Operation ID: `kv-v2-configure`

Configure backend level settings that are applied to every key in the key-value store.

Parameters

Parameter	Type	Location	Required	Description
`kv_v2_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`cas_required`	boolean	no	If true, the backend will require the cas parameter to be set for each write
`delete_version_after`	integer	no	If set, the length of time before a version is deleted. A negative duration disables the use of delete_version_after on all keys. A zero duration clears the current setting. Accepts a Go duration format string.
`max_versions`	integer	no	The number of versions to keep for each key. Defaults to 10

Responses

204: No Content

GET /{kv_v2_mount_path}/data/{path}

Operation ID: `kv-v2-read`

Write, Patch, Read, and Delete data in the Key-Value Store.

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	Location of the secret.
`kv_v2_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`data`	object	no	
`metadata`	object	no	

POST /{kv_v2_mount_path}/data/{path}

Operation ID: `kv-v2-write`

Write, Patch, Read, and Delete data in the Key-Value Store.

Creation supported: yes

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	Location of the secret.
`kv_v2_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`data`	object	no	The contents of the data map will be stored and returned on read.
`options`	object	no	Options for writing a KV entry. Set the “cas” value to use a Check-And-Set operation. If not set the write will be allowed. If set to 0 a write will only be allowed if the key doesn’t exist. If the index is non-zero the write will only be allowed if the key’s current version matches the version specified in the cas parameter.
`override_version`	integer	no	Only replication!!!!!!!!
`version`	integer	no	If provided during a read, the value at the version number will be returned

Responses

200: OK

Parameter	Type	Required	Description
`created_time`	string	no	
`custom_metadata`	object	no	
`deletion_time`	string	no	
`destroyed`	boolean	no	
`version`	integer	no	

DELETE /{kv_v2_mount_path}/data/{path}

Operation ID: `kv-v2-delete`

Write, Patch, Read, and Delete data in the Key-Value Store.

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	Location of the secret.
`kv_v2_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: No Content

POST /{kv_v2_mount_path}/delete/{path}

Operation ID: `kv-v2-delete-versions`

Marks one or more versions as deleted in the KV store.

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	Location of the secret.
`kv_v2_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`versions`	array	no	The versions to be archived. The versioned data will not be deleted, but it will no longer be returned in normal get requests.

Responses

204: No Content

POST /{kv_v2_mount_path}/destroy/{path}

Operation ID: `kv-v2-destroy-versions`

Permanently removes one or more versions in the KV store

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	Location of the secret.
`kv_v2_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`versions`	array	no	The versions to destroy. Their data will be permanently deleted.

Responses

204: No Content

GET /{kv_v2_mount_path}/metadata/{path}

Operation ID: `kv-v2-read-metadata`

Configures settings for the KV store

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	Location of the secret.
`kv_v2_mount_path`	string	path	yes	Path that the backend was mounted at
`list`	string	query	no	Return a list if `true`

Responses

200: OK

Parameter	Type	Required	Description
`cas_required`	boolean	no	
`created_time`	string	no	
`current_version`	integer	no	
`custom_metadata`	object	no	User-provided key-value pairs that are used to describe arbitrary and version-agnostic information about a secret.
`delete_version_after`	integer	no	The length of time before a version is deleted.
`last_sync_time`	string	no	
`max_versions`	integer	no	The number of versions to keep
`oldest_version`	integer	no	
`updated_time`	string	no	
`versions`	object	no	

POST /{kv_v2_mount_path}/metadata/{path}

Operation ID: `kv-v2-write-metadata`

Configures settings for the KV store

Creation supported: yes

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	Location of the secret.
`kv_v2_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`cas_required`	boolean	no	If true the key will require the cas parameter to be set on all write requests. If false, the backend's configuration will be used.
`custom_metadata`	object	no	User-provided key-value pairs that are used to describe arbitrary and version-agnostic information about a secret.
`delete_version_after`	integer	no	The length of time before a version is deleted. If not set, the backend's configured delete_version_after is used. Cannot be greater than the backend's delete_version_after. A zero duration clears the current setting. A negative duration will cause an error.
`max_versions`	integer	no	The number of versions to keep. If not set, the backend's configured max version is used.

Responses

204: No Content

DELETE /{kv_v2_mount_path}/metadata/{path}

Operation ID: `kv-v2-delete-metadata`

Configures settings for the KV store

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	Location of the secret.
`kv_v2_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: No Content

GET /{kv_v2_mount_path}/subkeys/{path}

Operation ID: `kv-v2-read-subkeys`

Read the structure of a secret entry from the Key-Value store with the values removed.

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	Location of the secret.
`kv_v2_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`metadata`	object	no	
`subkeys`	object	no	

POST /{kv_v2_mount_path}/undelete/{path}

Operation ID: `kv-v2-undelete-versions`

Undeletes one or more versions from the KV store.

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	Location of the secret.
`kv_v2_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`versions`	array	no	The versions to unarchive. The versions will be restored and their data will be returned on normal get requests.

Responses

204: No Content

GET /{ldap_mount_path}/config

Operation ID: `ldap-read-configuration`

Configure the LDAP secrets engine plugin.

Parameters

Parameter	Type	Location	Required	Description
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /{ldap_mount_path}/config

Operation ID: `ldap-configure`

Configure the LDAP secrets engine plugin.

Creation supported: yes

Parameters

Parameter	Type	Location	Required	Description
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`anonymous_group_search`	boolean (default: False)	no	Use anonymous binds LDAP group searches credentials will still be connection test).
`binddn`	string	no	LDAP DN for searchin (optional)
`bindpass`	string	no	LDAP password for se user DN (optional)
`case_sensitive_names`	boolean	no	If true, case sensitivity when comparing usern groups for matching po
`certificate`	string	no	CA certificate to use w LDAP server certificate PEM encoded (optiona
`client_tls_cert`	string	no	Client certificate to pro server, must be x509 F (optional)
`client_tls_key`	string	no	Client certificate key to LDAP server, must be encoded (optional)
`connection_timeout`	integer (default: 30s)	no	Timeout, in seconds, w to connect to the LDAP trying the next URL in t
`credential_type`	string (default: 1)	no	The type of credential Options include: ‘passw Defaults to ‘password’.
`deny_null_bind`	boolean (default: True)	no	⚠ Deprecated. Denies unauthenticated LDAP the user’s password is to true
`dereference_aliases`	string (never, finding, searching, always) (default: never)	no	When aliases should b on search operations. A are ’never’, ‘finding’, ‘s ‘always’. Defaults to ’n

Parameter	Type	Required	Description
`discoverdn`	boolean	no	Use anonymous bind t bind DN of a user (opti
`enable_samaccountname_login`	boolean (default: False)	no	If true, matching sAMA attribute values will be when upndomain is de
`groupattr`	string (default: cn)	no	LDAP attribute to follow returned by in order to group membership. Ex “memberOf”, etc. Defa
`groupdn`	string	no	LDAP search base to u membership search (e ou=Groups,dc=exampl
`groupfilter`	string (default: (| (memberUid={{.Username}}) (member={{.UserDN}}) (uniqueMember={{.UserDN}})))	no	Go template for queryi membership of user (o template can access th context variables: User Example: (&(objectCla (member: 1.2.840.113556.1.4.19 Default: (|(memberUid= (member={{.UserDN}}) (uniqueMember={{.Use
`insecure_tls`	boolean	no	Skip LDAP server SSL verification - VERY ins
`length`	integer	no	⚠ Deprecated. The de passwords that Vault g
`max_page_size`	integer (default: 0)	no	If set to a value greate LDAP backend will use server’s paged search request pages of up to This can be used to av LDAP server’s maximu limit. Otherwise, the LD not use the paged sea
`max_ttl`	integer	no	The maximum passwo

Parameter	Type	Required	Description
`password_policy`	string	no	Password policy to use passwords
`request_timeout`	integer (default: 90s)	no	Timeout, in seconds, fo when making requests server before returning
`schema`	string (default: openldap)	no	The desired LDAP sch modifying user accoun
`skip_static_role_import_rotation`	boolean	no	Whether to skip the 'im
`starttls`	boolean	no	Issue a StartTLS comm establishing unencrypt (optional)
`tls_max_version`	string (tls10, tls11, tls12, tls13) (default: tls12)	no	Maximum TLS version values are 'tls10', 'tls11 'tls13'. Defaults to 'tls1
`tls_min_version`	string (tls10, tls11, tls12, tls13) (default: tls12)	no	Minimum TLS version values are 'tls10', 'tls11 'tls13'. Defaults to 'tls1
`ttl`	integer	no	The default password t
`upndomain`	string	no	Enables userPrincipalD [username]@UPNDom
`url`	string (default: ldap://127.0.0.1)	no	LDAP URL to connect 127.0.0.1). Multiple UR specified by concatena commas; they will be tr
`use_pre111_group_cn_behavior`	boolean	no	In Vault 1.1.1 a fix for h CN values of different unfortunately introduce that could cause previo groups to not be found in the resulting name. pre-1.1.1 behavior for CNs will be used. This some upgrade scenari compatibility. It is enab

Parameter	Type	Required	Description
			the config is upgraded default on new configu
`use_token_groups`	boolean (default: False)	no	If true, use the Active D tokenGroups construct the user to find the gro This will find all securit including nested ones.
`userattr`	string (default: cn)	no	Attribute used for users
`userdn`	string	no	LDAP domain to use fo ou=People,dc=exampl
`userfilter`	string (default: ({{.UserAttr}} ={{.Username}}))	no	Go template for LDAP (optional) The template following context varial Username Default: ({{. ={{.Username}})
`username_as_alias`	boolean (default: False)	no	If true, sets the alias na username

Responses

200: OK

DELETE /{ldap_mount_path}/config

Operation ID: `ldap-delete-configuration`

Configure the LDAP secrets engine plugin.

Parameters

Parameter	Type	Location	Required	Description
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: empty body

GET /{ldap_mount_path}/creds/{name}

Operation ID: `ldap-request-dynamic-role-credentials`

Request LDAP credentials for a dynamic role. These credentials are created within the LDAP system when querying this endpoint.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the dynamic role.
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

GET /{ldap_mount_path}/library

Operation ID: `ldap-library-list`

List the name of each set of service accounts currently stored.

Parameters

Parameter	Type	Location	Required	Description
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

POST /{ldap_mount_path}/library/manage/{name}/check-in

Operation ID: `ldap-library-force-check-in`

Check service accounts in to the library.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the set.
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`service_account_names`	array	no	The username/logon name for the service accounts to check in.

Responses

200: OK

GET /{ldap_mount_path}/library/{name}

Operation ID: `ldap-library-read`

Read a library set.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the set.
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /{ldap_mount_path}/library/{name}

Operation ID: `ldap-library-configure`

Update a library set.

Creation supported: yes

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the set.
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`disable_check_in_enforcement`	boolean (default: False)	no	Disable the default behavior of requiring that check-ins are performed by the entity that checked them out.
`max_ttl`	integer (default: 86400)	no	In seconds, the max amount of time a check-out's renewals should last. Defaults to 24 hours.
`service_account_names`	array	no	The username/logon name for the service accounts with which this set will be associated.
`ttl`	integer (default: 86400)	no	In seconds, the amount of time a check-out should last. Defaults to 24 hours.

Responses

200: OK

DELETE /{ldap_mount_path}/library/{name}

Operation ID: `ldap-library-delete`

Delete a library set.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the set.
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: empty body

POST /{ldap_mount_path}/library/{name}/check-in

Operation ID: `ldap-library-check-in`

Check service accounts in to the library.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the set.
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`service_account_names`	array	no	The username/logon name for the service accounts to check in.

Responses

200: OK

POST /{ldap_mount_path}/library/{name}/check-out

Operation ID: `ldap-library-check-out`

Check a service account out from the library.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the set
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`ttl`	integer	no	The length of time before the check-out will expire, in seconds.

Responses

200: OK

GET /{ldap_mount_path}/library/{name}/status

Operation ID: `ldap-library-check-status`

Check the status of the service accounts in a library set.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the set.
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

GET /{ldap_mount_path}/library/{path}

Operation ID: `ldap-library-list-library-path`

List the name of each set of service accounts currently stored.

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	Path of sets to list
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /{ldap_mount_path}/library{path}

Operation ID: `ldap-library-list-library-path`

List the name of each set of service accounts currently stored.

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	Path of sets to list
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /{ldap_mount_path}/role

Operation ID: `ldap-list-dynamic-roles`

List all the dynamic roles Vault is currently managing in LDAP.

Parameters

Parameter	Type	Location	Required	Description
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /{ldap_mount_path}/role/{name}

Operation ID: `ldap-read-dynamic-role`

Manage the static roles that can be created with this backend.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the role (lowercase)
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /{ldap_mount_path}/role/{name}

Operation ID: `ldap-write-dynamic-role`

Manage the static roles that can be created with this backend.

Creation supported: yes

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the role (lowercase)
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`creation_ldif`	string	yes	LDIF string used to create new entities within the LDAP system. This LDIF can be templated.
`default_ttl`	integer	no	Default TTL for dynamic credentials
`deletion_ldif`	string	yes	LDIF string used to delete entities created within the LDAP system. This LDIF can be templated.
`max_ttl`	integer	no	Max TTL a dynamic credential can be extended to
`rollback_ldif`	string	no	LDIF string used to rollback changes in the event of a failure to create credentials. This LDIF can be templated.
`username_template`	string	no	The template used to create a username

Responses

200: OK

DELETE /{ldap_mount_path}/role/{name}

Operation ID: `ldap-delete-dynamic-role`

Manage the static roles that can be created with this backend.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the role (lowercase)
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: empty body

GET /{ldap_mount_path}/role/{path}

Operation ID: `ldap-list-role-path`

List all the dynamic roles Vault is currently managing in LDAP.

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	Path of roles to list
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /{ldap_mount_path}/role{path}

Operation ID: `ldap-list-role-path`

List all the dynamic roles Vault is currently managing in LDAP.

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	Path of roles to list
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

POST /{ldap_mount_path}/rotate-role/{name}

Operation ID: `ldap-rotate-static-role`

Request to rotate the credentials for a static user account.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the role
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`dn`	string	no	The distinguished name of the entry to manage.
`username`	string	no	The username/logon name for the entry with which this role will be associated.

Responses

200: OK

POST /{ldap_mount_path}/rotate-root

Operation ID: `ldap-rotate-root-credentials`

Request to rotate the root credentials Vault uses for the LDAP administrator account.

Parameters

Parameter	Type	Location	Required	Description
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

GET /{ldap_mount_path}/static-cred/{name}

Operation ID: `ldap-request-static-role-credentials`

Request LDAP credentials for a certain static role. These credentials are rotated periodically.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the static role.
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

GET /{ldap_mount_path}/static-role

Operation ID: `ldap-list-static-roles`

This path lists all the static roles Vault is currently managing within the LDAP system.

Parameters

Parameter	Type	Location	Required	Description
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /{ldap_mount_path}/static-role/{name}

Operation ID: `ldap-read-static-role`

Manage the static roles that can be created with this backend.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the role
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /{ldap_mount_path}/static-role/{name}

Operation ID: `ldap-write-static-role`

Manage the static roles that can be created with this backend.

Creation supported: yes

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the role
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`dn`	string	no	The distinguished name of the entry to manage.
`rotation_period`	integer	no	Period for automatic credential rotation of the given entry.
`skip_import_rotation`	boolean	no	Skip the initial pasword rotation on import (has no effect on updates)
`username`	string	no	The username/logon name for the entry with which this role will be associated.

Responses

200: OK

DELETE /{ldap_mount_path}/static-role/{name}

Operation ID: `ldap-delete-static-role`

Manage the static roles that can be created with this backend.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the role
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: empty body

GET /{ldap_mount_path}/static-role/{path}

Operation ID: `ldap-list-static-role-path`

This path lists all the static roles Vault is currently managing within the LDAP system.

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	Path of roles to list
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /{ldap_mount_path}/static-role{path}

Operation ID: `ldap-list-static-role-path`

This path lists all the static roles Vault is currently managing within the LDAP system.

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	Path of roles to list
`ldap_mount_path`	string	path	yes	Path that the backend was mounted at
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

POST /{pki_mount_path}/acme/account/{kid}

Operation ID: `pki-write-acme-account-kid`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`kid`	string	path	yes	The key identifier provided by the CA
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request ‘payload’ value
`protected`	string	no	ACME request ‘protected’ value
`signature`	string	no	ACME request ‘signature’ value

Responses

200: OK

POST /{pki_mount_path}/acme/authorization/{auth_id}

Operation ID: `pki-write-acme-authorization-auth_id`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`auth_id`	string	path	yes	ACME authorization identifier value
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request ‘payload’ value
`protected`	string	no	ACME request ‘protected’ value
`signature`	string	no	ACME request ‘signature’ value

Responses

200: OK

POST /{pki_mount_path}/acme/challenge/{auth_id}/{challenge_type}

Operation ID: `pki-write-acme-challenge-auth_id-challenge_type`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`auth_id`	string	path	yes	ACME authorization identifier value
`challenge_type`	string	path	yes	ACME challenge type
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request ‘payload’ value
`protected`	string	no	ACME request ‘protected’ value
`signature`	string	no	ACME request ‘signature’ value

Responses

200: OK

GET /{pki_mount_path}/acme/directory

Operation ID: `pki-read-acme-directory`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /{pki_mount_path}/acme/new-account

Operation ID: `pki-write-acme-new-account`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request ‘payload’ value
`protected`	string	no	ACME request ‘protected’ value
`signature`	string	no	ACME request ‘signature’ value

Responses

200: OK

POST /{pki_mount_path}/acme/new-eab

Operation ID: `pki-generate-eab-key`

Generate external account bindings to be used for ACME

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`acme_directory`	string	no	The ACME directory to which the key belongs
`created_on`	string	no	An RFC3339 formatted date time when the EAB token was created
`id`	string	no	The EAB key identifier
`key`	string	no	The EAB hmac key
`key_type`	string	no	The EAB key type

GET /{pki_mount_path}/acme/new-nonce

Operation ID: `pki-read-acme-new-nonce`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /{pki_mount_path}/acme/new-order

Operation ID: `pki-write-acme-new-order`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request ‘payload’ value
`protected`	string	no	ACME request ‘protected’ value
`signature`	string	no	ACME request ‘signature’ value

Responses

200: OK

POST /{pki_mount_path}/acme/order/{order_id}

Operation ID: `pki-write-acme-order-order_id`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`order_id`	string	path	yes	The ACME order identifier to fetch
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request ‘payload’ value
`protected`	string	no	ACME request ‘protected’ value
`signature`	string	no	ACME request ‘signature’ value

Responses

200: OK

POST /{pki_mount_path}/acme/order/{order_id}/cert

Operation ID: `pki-write-acme-order-order_id-cert`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`order_id`	string	path	yes	The ACME order identifier to fetch
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request ‘payload’ value
`protected`	string	no	ACME request ‘protected’ value
`signature`	string	no	ACME request ‘signature’ value

Responses

200: OK

POST /{pki_mount_path}/acme/order/{order_id}/finalize

Operation ID: `pki-write-acme-order-order_id-finalize`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`order_id`	string	path	yes	The ACME order identifier to fetch
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request ‘payload’ value
`protected`	string	no	ACME request ‘protected’ value
`signature`	string	no	ACME request ‘signature’ value

Responses

200: OK

POST /{pki_mount_path}/acme/orders

Operation ID: `pki-write-acme-orders`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request ‘payload’ value
`protected`	string	no	ACME request ‘protected’ value
`signature`	string	no	ACME request ‘signature’ value

Responses

200: OK

POST /{pki_mount_path}/acme/revoke-cert

Operation ID: `pki-write-acme-revoke-cert`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request ‘payload’ value
`protected`	string	no	ACME request ‘protected’ value
`signature`	string	no	ACME request ‘signature’ value

Responses

200: OK

GET /{pki_mount_path}/ca

Operation ID: `pki-read-ca-der`

Fetch a CA, CRL, CA Chain, or non-revoked certificate.

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`ca_chain`	string	no	Issuing CA Chain
`certificate`	string	no	Certificate
`issuer_id`	string	no	ID of the issuer
`revocation_time`	integer	no	Revocation time
`revocation_time_rfc3339`	string	no	Revocation time RFC 3339 formatted

GET /{pki_mount_path}/ca/pem

Operation ID: `pki-read-ca-pem`

Fetch a CA, CRL, CA Chain, or non-revoked certificate.

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`ca_chain`	string	no	Issuing CA Chain
`certificate`	string	no	Certificate
`issuer_id`	string	no	ID of the issuer
`revocation_time`	integer	no	Revocation time
`revocation_time_rfc3339`	string	no	Revocation time RFC 3339 formatted

GET /{pki_mount_path}/ca_chain

Operation ID: `pki-read-ca-chain-pem`

Fetch a CA, CRL, CA Chain, or non-revoked certificate.

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`ca_chain`	string	no	Issuing CA Chain
`certificate`	string	no	Certificate
`issuer_id`	string	no	ID of the issuer
`revocation_time`	integer	no	Revocation time
`revocation_time_rfc3339`	string	no	Revocation time RFC 3339 formatted

GET /{pki_mount_path}/cert/ca_chain

Operation ID: `pki-read-cert-ca-chain`

Fetch a CA, CRL, CA Chain, or non-revoked certificate.

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`ca_chain`	string	no	Issuing CA Chain
`certificate`	string	no	Certificate
`issuer_id`	string	no	ID of the issuer
`revocation_time`	integer	no	Revocation time
`revocation_time_rfc3339`	string	no	Revocation time RFC 3339 formatted

GET /{pki_mount_path}/cert/crl

Operation ID: `pki-read-cert-crl`

Fetch a CA, CRL, CA Chain, or non-revoked certificate.

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`ca_chain`	string	no	Issuing CA Chain
`certificate`	string	no	Certificate
`issuer_id`	string	no	ID of the issuer
`revocation_time`	integer	no	Revocation time
`revocation_time_rfc3339`	string	no	Revocation time RFC 3339 formatted

GET /{pki_mount_path}/cert/delta-crl

Operation ID: `pki-read-cert-delta-crl`

Fetch a CA, CRL, CA Chain, or non-revoked certificate.

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`ca_chain`	string	no	Issuing CA Chain
`certificate`	string	no	Certificate
`issuer_id`	string	no	ID of the issuer
`revocation_time`	integer	no	Revocation time
`revocation_time_rfc3339`	string	no	Revocation time RFC 3339 formatted

GET /{pki_mount_path}/cert/unified-crl

Operation ID: `pki-read-cert-unified-crl`

Fetch a CA, CRL, CA Chain, or non-revoked certificate.

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`ca_chain`	string	no	Issuing CA Chain
`certificate`	string	no	Certificate
`issuer_id`	string	no	ID of the issuer
`revocation_time`	integer	no	Revocation time
`revocation_time_rfc3339`	string	no	Revocation time RFC 3339 formatted

GET /{pki_mount_path}/cert/unified-delta-crl

Operation ID: `pki-read-cert-unified-delta-crl`

Fetch a CA, CRL, CA Chain, or non-revoked certificate.

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`ca_chain`	string	no	Issuing CA Chain
`certificate`	string	no	Certificate
`issuer_id`	string	no	ID of the issuer
`revocation_time`	integer	no	Revocation time
`revocation_time_rfc3339`	string	no	Revocation time RFC 3339 formatted

GET /{pki_mount_path}/cert/{serial}

Operation ID: `pki-read-cert`

Fetch a CA, CRL, CA Chain, or non-revoked certificate.

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`serial`	string	path	yes	Certificate serial number, in colon- or hyphen-separated octal
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`ca_chain`	string	no	Issuing CA Chain
`certificate`	string	no	Certificate
`issuer_id`	string	no	ID of the issuer
`revocation_time`	integer	no	Revocation time
`revocation_time_rfc3339`	string	no	Revocation time RFC 3339 formatted

GET /{pki_mount_path}/cert/{serial}/raw

Operation ID: `pki-read-cert-raw-der`

Fetch a CA, CRL, CA Chain, or non-revoked certificate.

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`serial`	string	path	yes	Certificate serial number, in colon- or hyphen-separated octal
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`ca_chain`	string	no	Issuing CA Chain
`certificate`	string	no	Certificate
`issuer_id`	string	no	ID of the issuer
`revocation_time`	integer	no	Revocation time
`revocation_time_rfc3339`	string	no	Revocation time RFC 3339 formatted

GET /{pki_mount_path}/cert/{serial}/raw/pem

Operation ID: `pki-read-cert-raw-pem`

Fetch a CA, CRL, CA Chain, or non-revoked certificate.

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`serial`	string	path	yes	Certificate serial number, in colon- or hyphen-separated octal
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`ca_chain`	string	no	Issuing CA Chain
`certificate`	string	no	Certificate
`issuer_id`	string	no	ID of the issuer
`revocation_time`	integer	no	Revocation time
`revocation_time_rfc3339`	string	no	Revocation time RFC 3339 formatted

GET /{pki_mount_path}/certs

Operation ID: `pki-list-certs`

Fetch a CA, CRL, CA Chain, or non-revoked certificate.

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

Parameter	Type	Required	Description
`keys`	array	no	A list of keys

GET /{pki_mount_path}/certs/revocation-queue

Operation ID: `pki-list-certs-revocation-queue`

List all pending, cross-cluster revocations known to the local cluster.

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /{pki_mount_path}/certs/revoked

Operation ID: `pki-list-revoked-certs`

List all revoked serial numbers within the local cluster

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

Parameter	Type	Required	Description
`keys`	array	no	List of Keys

GET /{pki_mount_path}/certs/unified-revoked

Operation ID: `pki-list-unified-revoked-certs`

List all revoked serial numbers within this cluster’s unified storage area.

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

Parameter	Type	Required	Description
`key_info`	string	no	Key information
`keys`	array	no	List of Keys

GET /{pki_mount_path}/config/acme

Operation ID: `pki-read-acme-configuration`

Configuration of ACME Endpoints

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /{pki_mount_path}/config/acme

Operation ID: `pki-configure-acme`

Configuration of ACME Endpoints

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`allow_role_ext_key_usage`	boolean (default: False)	no	whether the ExtKeyUsage field from a role is used, defaults to false meaning that certificate will be signed with ServerAuth.
`allowed_issuers`	array (default: ['*'])	no	which issuers are allowed for use with ACME; by default, this will only be the primary (default) issuer
`allowed_roles`	array (default: ['*'])	no	which roles are allowed for use with ACME; by default via '*', these will be all roles including sign-verbatim; when concrete role names are specified, any default_directory_policy role must be included to allow usage of the default acme directories under /pki/acme/ directory and /pki/issuer/:issuer_id/acme/ directory.
`default_directory_policy`	string (default: sign-verbatim)	no	the policy to be used for non-role-qualified ACME requests; by default ACME issuance will be otherwise unrestricted, equivalent to the sign-verbatim endpoint; one may also specify a role to use as this policy, as "role:<role_name>", the specified role must be allowed by allowed_roles
`dns_resolver`	string (default:)	no	DNS resolver to use for domain resolution on this mount. Defaults to using the default system resolver. Must be in the format :, with both parts mandatory.
`eab_policy`	string (default: always-required)	no	Specify the policy to use for external account binding behaviour, 'not-required', 'new-account-required' or 'always-required'
`enabled`	boolean (default: False)	no	whether ACME is enabled, defaults to false meaning that clusters will by default not get ACME support

Parameter	Type	Required	Description
`max_ttl`	integer (default: 7776000)	no	maximum lifetime of ACME-issued certificates. This caps any role, mount, or issuer-derived certificate lifetime for ACME issuance and defaults to 2160h (90 days).

Responses

200: OK

GET /{pki_mount_path}/config/auto-tidy

Operation ID: `pki-read-auto-tidy-configuration`

Modifies the current configuration for automatic tidy execution.

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`acme_account_safety_buffer`	integer	no	Safety buffer after creation after which accounts lacking orders are revoked
`enabled`	boolean	no	Specifies whether automatic tidy is enabled or not
`interval_duration`	integer	no	Specifies the duration between automatic tidy operation
`issuer_safety_buffer`	integer	no	Issuer safety buffer
`maintain_stored_certificate_counts`	boolean	no	
`pause_duration`	string	no	Duration to pause between tidying certificates
`publish_stored_certificate_count_metrics`	boolean	no	
`revocation_queue_safety_buffer`	integer	no	
`safety_buffer`	integer	no	Safety buffer time duration
`tidy_acme`	boolean	no	Tidy Unused Acme Accounts, and Orders
`tidy_cert_store`	boolean	no	Specifies whether to tidy up the certificate store
`tidy_cross_cluster_revoked_certs`	boolean	no	
`tidy_expired_issuers`	boolean	no	Specifies whether tidy expired issuers
`tidy_move_legacy_ca_bundle`	boolean	no	
`tidy_revocation_queue`	boolean	no	
`tidy_revoked_cert_issuer_associations`	boolean	no	Specifies whether to associate revoked

Parameter	Type	Required	Description
			certificates with their corresponding issuers
`tidy_revoked_certs`	boolean	no	Specifies whether to remove all invalid and expired certificates from storage

POST /{pki_mount_path}/config/auto-tidy

Operation ID: `pki-configure-auto-tidy`

Modifies the current configuration for automatic tidy execution.

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`acme_account_safety_buffer`	integer (default: 2592000)	no	The amount of time that must pass after creation that an account with no orders is marked revoked, and the amount of time after being marked revoked or deactivated.
`enabled`	boolean	no	Set to true to enable automatic tidy operations.
`interval_duration`	integer (default: 43200)	no	Interval at which to run an auto-tidy operation. This is the time between tidy invocations (after one finishes to the start of the next). Running a manual tidy will reset this duration.
`issuer_safety_buffer`	integer (default: 31536000)	no	The amount of extra time that must have passed beyond issuer’s expiration before it is removed from the backend storage. Defaults to 8760 hours (1 year).
`maintain_stored_certificate_counts`	boolean (default: False)	no	This configures whether stored certificates are counted upon initialization of the backend, and whether during normal operation, a running count of certificates stored is maintained.
`pause_duration`	string (default: 0s)	no	The amount of time to wait between processing certificates. This allows operators to change the execution profile of tidy to take consume less resources by slowing down how long it takes to run.

Parameter	Type	Required	Description
			Note that the entire list of certificates will be stored in memory during the entire tidy operation, but resources to read/process/update existing entries will be spread out over a greater period of time. By default this is zero seconds.
`publish_stored_certificate_count_metrics`	boolean (default: False)	no	This configures whether the stored certificate count is published to the metrics consumer. It does not affect if the stored certificate count is maintained, and if maintained, it will be available on the tidy-status endpoint.
`revocation_queue_safety_buffer`	integer (default: 172800)	no	The amount of time that must pass from the cross-cluster revocation request being initiated to when it will be slated for removal. Setting this too low may remove valid revocation requests before the owning cluster has a chance to process them, especially if the cluster is offline.
`safety_buffer`	integer (default: 259200)	no	The amount of extra time that must have passed beyond certificate expiration before it is removed from the backend storage and/or revocation list. Defaults to 72 hours.
`tidy_acme`		no	Set to true to enable tidying ACME accounts, orders and

Parameter	Type	Required	Description
	boolean (default: False)		authorizations. ACME orders are tidied (deleted) safety_buffer after the certificate associated with them expires, or after the order and relevant authorizations have expired if no certificate was produced. Authorizations are tidied with the corresponding order. When a valid ACME Account is at least acme_account_safety_buffer old, and has no remaining orders associated with it, the account is marked as revoked. After another acme_account_safety_buffer has passed from the revocation or deactivation date, a revoked or deactivated ACME account is deleted.
`tidy_cert_store`	boolean	no	Set to true to enable tidying up the certificate store
`tidy_cross_cluster_revoked_certs`	boolean	no	Set to true to enable tidying up the cross-cluster revoked certificate store. Only runs on the active primary node.
`tidy_expired_issuers`	boolean	no	Set to true to automatically remove expired issuers past the issuer_safety_buffer. No keys will be removed as part of this operation.
`tidy_move_legacy_ca_bundle`	boolean	no	Set to true to move the legacy ca_bundle from /config/ca_bundle to /config/ca_bundle.bak. This

Parameter	Type	Required	Description
			prevents downgrades to pre-Vault 1.11 versions (as older PKI engines do not know about the new multi-issuer storage layout), but improves the performance on seal wrapped PKI mounts. This will only occur if at least issuer_safety_buffer time has occurred after the initial storage migration. This backup is saved in case of an issue in future migrations. Operators may consider removing it via sys/raw if they desire. The backup will be removed via a DELETE / root call, but note that this removes ALL issuers within the mount (and is thus not desirable in most operational scenarios).
`tidy_revocation_list`	boolean	no	Deprecated; synonym for 'tidy_revoked_certs
`tidy_revocation_queue`	boolean (default: False)	no	Set to true to remove stale revocation queue entries that haven't been confirmed by any active cluster. Only runs on the active primary node
`tidy_revoked_cert_issuer_associations`	boolean	no	Set to true to validate issuer associations on revocation entries. This helps increase the performance of CRL building and OCSP responses.
`tidy_revoked_certs`	boolean	no	

Parameter	Type	Required	Description
			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.

Responses

200: OK

Parameter	Type	Required	Description
`acme_account_safety_buffer`	integer	no	Safety buffer after creation after which accounts lacking orders are revoked
`enabled`	boolean	no	Specifies whether automatic tidy is enabled or not
`interval_duration`	integer	no	Specifies the duration between automatic tidy operation
`issuer_safety_buffer`	integer	no	Issuer safety buffer
`maintain_stored_certificate_counts`	boolean	no	
`pause_duration`	string	no	Duration to pause between tidying certificates
`publish_stored_certificate_count_metrics`	boolean	no	
`revocation_queue_safety_buffer`	integer	no	
`safety_buffer`	integer	no	Safety buffer time duration
`tidy_acme`	boolean	no	Tidy Unused Acme Accounts, and Orders
`tidy_cert_store`	boolean	no	Specifies whether to tidy up the certificate store
`tidy_cross_cluster_revoked_certs`	boolean	no	Tidy the cross-cluster revoked certificate store
`tidy_expired_issuers`	boolean	no	Specifies whether tidy expired issuers
`tidy_move_legacy_ca_bundle`	boolean	no	
`tidy_revocation_queue`	boolean	no	
`tidy_revoked_cert_issuer_associations`	boolean	no	Specifies whether to associate revoked

Parameter	Type	Required	Description
			certificates with their corresponding issuers
`tidy_revoked_certs`	boolean	no	Specifies whether to remove all invalid and expired certificates from storage

POST /{pki_mount_path}/config/ca

Operation ID: `pki-configure-ca`

Set the CA certificate and private key used for generated credentials.

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`pem_bundle`	string	no	PEM-format, concatenated unencrypted secret key and certificate.

Responses

200: OK

Parameter	Type	Required	Description
`existing_issuers`	array	no	Existing issuers specified as part of the import bundle of this request
`existing_keys`	array	no	Existing keys specified as part of the import bundle of this request
`imported_issuers`	array	no	Net-new issuers imported as a part of this request
`imported_keys`	array	no	Net-new keys imported as a part of this request
`mapping`	object	no	A mapping of issuer_id to key_id for all issuers included in this request

GET /{pki_mount_path}/config/cluster

Operation ID: `pki-read-cluster-configuration`

Set cluster-local configuration, including address to this PR cluster.

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`aia_path`	string	no	Optional URI to this mount's AIA distribution point; may refer to an external non-Vault responder. This is for resolving AIA URLs and providing the {{cluster_aia_path}} template parameter and will not be used for other purposes. As such, unlike path above, this could safely be an insecure transit mechanism (like HTTP without TLS). For example: http://cdn.example.com/pr1/pki
`path`	string	no	Canonical URI to this mount on this performance replication cluster's external address. This is for resolving AIA URLs and providing the {{cluster_path}} template parameter but might be used for other purposes in the future. This should only point back to this particular PR replica and should not ever point to another PR cluster. It may point to any node in the PR replica, including standby nodes, and need not always point to the active node. For example: https://pr1.vault.example.com:8200/v1/pki

POST /{pki_mount_path}/config/cluster

Operation ID: `pki-configure-cluster`

Set cluster-local configuration, including address to this PR cluster.

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`aia_path`	string	no	Optional URI to this mount's AIA distribution point; may refer to an external non-Vault responder. This is for resolving AIA URLs and providing the {{cluster_aia_path}} template parameter and will not be used for other purposes. As such, unlike path above, this could safely be an insecure transit mechanism (like HTTP without TLS). For example: http://cdn.example.com/pr1/pki
`path`	string	no	Canonical URI to this mount on this performance replication cluster's external address. This is for resolving AIA URLs and providing the {{cluster_path}} template parameter but might be used for other purposes in the future. This should only point back to this particular PR replica and should not ever point to another PR cluster. It may point to any node in the PR replica, including standby nodes, and need not always point to the active node. For example: https://pr1.vault.example.com:8200/v1/pki

Responses

200: OK

Parameter	Type	Required	Description
`aia_path`	string	no	Optional URI to this mount's AIA distribution point; may refer to an external non-Vault responder. This is for resolving AIA URLs and providing the {{cluster_aia_path}} template parameter and will not be used for other purposes. As such, unlike path above, this could safely be an insecure transit mechanism (like HTTP without TLS). For example: http://cdn.example.com/pr1/pki
`path`	string	no	Canonical URI to this mount on this performance replication cluster's external address. This is for resolving AIA URLs and providing the {{cluster_path}} template parameter but might be used for other purposes in the future. This should only point back to this particular PR replica and should not ever point to another PR cluster. It may point to any node in the PR replica, including standby nodes, and need not always point to the active node. For example: https://pr1.vault.example.com:8200/v1/pki

GET /{pki_mount_path}/config/crl

Operation ID: `pki-read-crl-configuration`

Configure the CRL expiration.

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`auto_rebuild`	boolean	no	If set to true, enables automatic rebuilding of the CRL
`auto_rebuild_grace_period`	string	no	The time before the CRL expires to automatically rebuild it, when enabled. Must be shorter than the CRL expiry. Defaults to 12h.
`cross_cluster_revocation`	boolean	no	Whether to enable a global, cross-cluster revocation queue. Must be used with auto_rebuild=true.
`delta_rebuild_interval`	string	no	The time between delta CRL rebuilds if a new revocation has occurred. Must be shorter than the CRL expiry. Defaults to 15m.
`disable`	boolean	no	If set to true, disables generating the CRL entirely.
`enable_delta`	boolean	no	Whether to enable delta CRLs between authoritative CRL rebuilds
`expiry`	string	no	The amount of time the generated CRL should be valid; defaults to 72 hours
`ocsp_disable`	boolean	no	If set to true, ocsp unauthorized responses will be returned.
`ocsp_expiry`	string	no	The amount of time an OCSP response will be valid (controls the NextUpdate field); defaults to 12 hours
`unified_crl`	boolean	no	If set to true enables global replication of revocation entries, also enabling unified versions of OCSP and CRLs if their respective features are enabled. disable for CRLs and ocsp_disable for OCSP.
`unified_crl_on_existing_paths`	boolean	no	If set to true, existing CRL and OCSP paths will return the unified CRL

Parameter	Type	Required	Description
			instead of a response based on cluster-local data

POST /{pki_mount_path}/config/crl

Operation ID: `pki-configure-crl`

Configure the CRL expiration.

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`auto_rebuild`	boolean	no	If set to true, enables automatic rebuilding of the CRL
`auto_rebuild_grace_period`	string (default: 12h)	no	The time before the CRL expires to automatically rebuild it, when enabled. Must be shorter than the CRL expiry. Defaults to 12h.
`cross_cluster_revocation`	boolean	no	Whether to enable a global, cross-cluster revocation queue. Must be used with auto_rebuild=true.
`delta_rebuild_interval`	string (default: 15m)	no	The time between delta CRL rebuilds if a new revocation has occurred. Must be shorter than the CRL expiry. Defaults to 15m.
`disable`	boolean	no	If set to true, disables generating the CRL entirely.
`enable_delta`	boolean	no	Whether to enable delta CRLs between authoritative CRL rebuilds
`expiry`	string (default: 72h)	no	The amount of time the generated CRL should be valid; defaults to 72 hours
`ocsp_disable`	boolean	no	If set to true, ocsp unauthorized responses will be returned.
`ocsp_expiry`	string (default: 1h)	no	The amount of time an OCSP response will be valid (controls the NextUpdate field); defaults to 12 hours
`unified_crl`	boolean (default: false)	no	If set to true enables global replication of revocation entries, also enabling unified versions of OCSP and CRLs if their respective features are enabled. disable for CRLs and ocsp_disable for OCSP.
`unified_crl_on_existing_paths`		no	If set to true, existing CRL and OCSP paths will return the unified

Parameter	Type	Required	Description
	boolean (default: false)		CRL instead of a response based on cluster-local data

Responses

200: OK

Parameter	Type	Required	Description
`auto_rebuild`	boolean	no	If set to true, enables automatic rebuilding of the CRL
`auto_rebuild_grace_period`	string (default: 12h)	no	The time before the CRL expires to automatically rebuild it, when enabled. Must be shorter than the CRL expiry. Defaults to 12h.
`cross_cluster_revocation`	boolean	no	Whether to enable a global, cross-cluster revocation queue. Must be used with auto_rebuild=true.
`delta_rebuild_interval`	string (default: 15m)	no	The time between delta CRL rebuilds if a new revocation has occurred. Must be shorter than the CRL expiry. Defaults to 15m.
`disable`	boolean	no	If set to true, disables generating the CRL entirely.
`enable_delta`	boolean	no	Whether to enable delta CRLs between authoritative CRL rebuilds
`expiry`	string (default: 72h)	no	The amount of time the generated CRL should be valid; defaults to 72 hours
`ocsp_disable`	boolean	no	If set to true, ocsp unauthorized responses will be returned.
`ocsp_expiry`	string (default: 1h)	no	The amount of time an OCSP response will be valid (controls the NextUpdate field); defaults to 12 hours
`unified_crl`	boolean	no	If set to true enables global replication of revocation entries, also enabling unified versions of OCSP and CRLs if their respective features are enabled. disable for CRLs and ocsp_disable for OCSP.
`unified_crl_on_existing_paths`	boolean	no	If set to true, existing CRL and OCSP paths will return the unified

Parameter	Type	Required	Description
			CRL instead of a response based on cluster-local data

GET /{pki_mount_path}/config/issuers

Operation ID: `pki-read-issuers-configuration`

Read and set the default issuer certificate for signing.

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`default`	string	no	Reference (name or identifier) to the default issuer.
`default_follows_latest_issuer`	boolean	no	Whether the default issuer should automatically follow the latest generated or imported issuer. Defaults to false.

POST /{pki_mount_path}/config/issuers

Operation ID: `pki-configure-issuers`

Read and set the default issuer certificate for signing.

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`default`	string	no	Reference (name or identifier) to the default issuer.
`default_follows_latest_issuer`	boolean (default: False)	no	Whether the default issuer should automatically follow the latest generated or imported issuer. Defaults to false.

Responses

200: OK

Parameter	Type	Required	Description
`default`	string	no	Reference (name or identifier) to the default issuer.
`default_follows_latest_issuer`	boolean	no	Whether the default issuer should automatically follow the latest generated or imported issuer. Defaults to false.

GET /{pki_mount_path}/config/keys

Operation ID: `pki-read-keys-configuration`

Read and set the default key used for signing

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`default`	string	no	Reference (name or identifier) to the default issuer.

POST /{pki_mount_path}/config/keys

Operation ID: `pki-configure-keys`

Read and set the default key used for signing

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`default`	string	no	Reference (name or identifier) of the default key.

Responses

200: OK

Parameter	Type	Required	Description
`default`	string	no	Reference (name or identifier) to the default issuer.

GET /{pki_mount_path}/config/urls

Operation ID: `pki-read-urls-configuration`

Set the URLs for the issuing CA, CRL distribution points, and OCSP servers.

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`crl_distribution_points`	array	no	Comma-separated list of URLs to be used for the CRL distribution points attribute. See also RFC 5280 Section 4.2.1.13.
`enable_templating`	boolean	no	Whether or not to enable templating of the above AIA fields. When templating is enabled the special values '{{issuer_id}}' and '{{cluster_path}}' are available, but the addresses are not checked for URI validity until issuance time. This requires /config/cluster's path to be set on all PR Secondary clusters.
`issuing_certificates`	array	no	Comma-separated list of URLs to be used for the issuing certificate attribute. See also RFC 5280 Section 4.2.2.1.
`ocsp_servers`	array	no	Comma-separated list of URLs to be used for the OCSP servers attribute. See also RFC 5280 Section 4.2.2.1.

POST /{pki_mount_path}/config/urls

Operation ID: `pki-configure-urls`

Set the URLs for the issuing CA, CRL distribution points, and OCSP servers.

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`crl_distribution_points`	array	no	Comma-separated list of URLs to be used for the CRL distribution points attribute. See also RFC 5280 Section 4.2.1.13.
`enable_templating`	boolean (default: False)	no	Whether or not to enabling templating of the above AIA fields. When templating is enabled the special values ‘{{issuer_id}}’, ‘{{cluster_path}}’, and ‘{{cluster_aia_path}}’ are available, but the addresses are not checked for URI validity until issuance time. Using ‘{{cluster_path}}’ requires /config/cluster’s ‘path’ member to be set on all PR Secondary clusters and using ‘{{cluster_aia_path}}’ requires /config/cluster’s ‘aia_path’ member to be set on all PR secondary clusters.
`issuing_certificates`	array	no	Comma-separated list of URLs to be used for the issuing certificate attribute. See also RFC 5280 Section 4.2.2.1.
`ocsp_servers`	array	no	Comma-separated list of URLs to be used for the OCSP servers attribute. See also RFC 5280 Section 4.2.2.1.

Responses

200: OK

Parameter	Type	Required	Description
`crl_distribution_points`	array	no	Comma-separated list of URLs to be used for the CRL distribution points attribute. See also RFC 5280 Section 4.2.1.13.
`enable_templating`	boolean (default: False)	no	Whether or not to enabling templating of the above AIA fields. When templating is enabled the special values '{{issuer_id}}' and '{{cluster_path}}' are available, but the addresses are not checked for URI validity until issuance time. This requires /config/cluster's path to be set on all PR Secondary clusters.
`issuing_certificates`	array	no	Comma-separated list of URLs to be used for the issuing certificate attribute. See also RFC 5280 Section 4.2.2.1.
`ocsp_servers`	array	no	Comma-separated list of URLs to be used for the OCSP servers attribute. See also RFC 5280 Section 4.2.2.1.

GET /{pki_mount_path}/crl

Operation ID: `pki-read-crl-der`

Fetch a CA, CRL, CA Chain, or non-revoked certificate.

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`ca_chain`	string	no	Issuing CA Chain
`certificate`	string	no	Certificate
`issuer_id`	string	no	ID of the issuer
`revocation_time`	integer	no	Revocation time
`revocation_time_rfc3339`	string	no	Revocation time RFC 3339 formatted

GET /{pki_mount_path}/crl/delta

Operation ID: `pki-read-crl-delta`

Fetch a CA, CRL, CA Chain, or non-revoked certificate.

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`ca_chain`	string	no	Issuing CA Chain
`certificate`	string	no	Certificate
`issuer_id`	string	no	ID of the issuer
`revocation_time`	integer	no	Revocation time
`revocation_time_rfc3339`	string	no	Revocation time RFC 3339 formatted

GET /{pki_mount_path}/crl/delta/pem

Operation ID: `pki-read-crl-delta-pem`

Fetch a CA, CRL, CA Chain, or non-revoked certificate.

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`ca_chain`	string	no	Issuing CA Chain
`certificate`	string	no	Certificate
`issuer_id`	string	no	ID of the issuer
`revocation_time`	integer	no	Revocation time
`revocation_time_rfc3339`	string	no	Revocation time RFC 3339 formatted

GET /{pki_mount_path}/crl/pem

Operation ID: `pki-read-crl-pem`

Fetch a CA, CRL, CA Chain, or non-revoked certificate.

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`ca_chain`	string	no	Issuing CA Chain
`certificate`	string	no	Certificate
`issuer_id`	string	no	ID of the issuer
`revocation_time`	integer	no	Revocation time
`revocation_time_rfc3339`	string	no	Revocation time RFC 3339 formatted

GET /{pki_mount_path}/crl/rotate

Operation ID: `pki-rotate-crl`

Force a rebuild of the CRL.

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`success`	boolean	no	Whether rotation was successful

GET /{pki_mount_path}/crl/rotate-delta

Operation ID: `pki-rotate-delta-crl`

Force a rebuild of the delta CRL.

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`success`	boolean	no	Whether rotation was successful

GET /{pki_mount_path}/eab

Operation ID: `pki-list-eab-keys`

list external account bindings to be used for ACME

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

Parameter	Type	Required	Description
`key_info`	object	no	EAB details keyed by the eab key id
`keys`	array	no	A list of unused eab keys

DELETE /{pki_mount_path}/eab/{key_id}

Operation ID: `pki-delete-eab-key`

Delete an external account binding id prior to its use within an ACME account

Parameters

Parameter	Type	Location	Required	Description
`key_id`	string	path	yes	EAB key identifier
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: empty body

POST /{pki_mount_path}/intermediate/cross-sign

Operation ID: `pki-cross-sign-intermediate`

Generate a new CSR and private key used for signing.

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`add_basic_constraints`	boolean	no	Whether to add a Basic Constraints extension with CA: true. Only needed as a workaround in some compatibility scenarios with Active Directory Certificate Services.
`alt_names`	string	no	The requested Subject Alternative Names, if any, in a comma-delimited list. May contain both DNS names and email addresses.
`common_name`	string	no	The requested common name; if you want more than one, specify the alternative names in the alt_names map. If not specified when signing, the common name will be taken from the CSR; other names must still be specified in alt_names or ip_sans.
`country`	array	no	If set, Country will be set to this value.
`exclude_cn_from_sans`	boolean (default: False)	no	If true, the Common Name will not be included in DNS or Email Subject Alternate Names. Defaults to false (CN is included).
`exported`	string (internal, exported, kms)	no	Must be “internal”, “exported” or “kms”. If set to “exported”, the generated private key will be returned. This is your *only* chance to retrieve the private key!
`format`	string (pem, der, pem_bundle) (default: pem)	no	Format for returned data. Can be “pem”, “der”, or “pem_bundle”. If “pem_bundle”, any private key

Parameter	Type	Required	Description
			and issuing cert will be appended to the certificate pem. If “der”, the value will be base64 encoded. Defaults to “pem”.
`ip_sans`	array	no	The requested IP SANs, if any, in a comma-delimited list
`key_bits`	integer (default: 0)	no	The number of bits to use. Allowed values are 0 (universal default); with rsa key_type: 2048 (default), 3072, or 4096; with ec key_type: 224, 256 (default), 384, or 521; ignored with ed25519.
`key_name`	string	no	Provide a name to the generated or existing key, the name must be unique across all keys and not be the reserved value ‘default’
`key_ref`	string (default: default)	no	Reference to a existing key; either “default” for the configured default key, an identifier or the name assigned to the key.
`key_type`	string (rsa, ec, ed25519, gost3410-256-paramset-a, gost3410-256-paramset-b, gost3410-256-paramset-c, gost3410-256-paramset-d, gost3410-512-paramset-a, gost3410-512-paramset-b, gost3410-512-	no	The type of key to use; defaults to RSA. “rsa” “ec”, “ed25519”, “gost3410-256-paramset-a”, “gost3410-256-paramset-b”, “gost3410-256-paramset-c”, “gost3410-256-paramset-d”, “gost3410-512-paramset-a”, “gost3410-512-paramset-b”, “gost3410-512-paramset-c” are the only valid values.

Parameter	Type	Required	Description
	paramset-c) (default: rsa)		
`locality`	array	no	If set, Locality will be set to this value.
`managed_key_id`	string	no	The name of the managed key to use when the exported type is kms. When kms type is the key type, this field or managed_key_name is required. Ignored for other types.
`managed_key_name`	string	no	The name of the managed key to use when the exported type is kms. When kms type is the key type, this field or managed_key_id is required. Ignored for other types.
`not_after`	string	no	Set the not after field of the certificate with specified date value. The value format should be given in UTC format YYYY-MM-ddTHH:MM:SSZ
`not_before_duration`	integer (default: 30)	no	The duration before now which the certificate needs to be backdated by.
`organization`	array	no	If set, O (Organization) will be set to this value.
`other_sans`	array	no	Requested other SANs, in an array with the format ;UTF8: for each entry.
`ou`	array	no	If set, OU (OrganizationalUnit) will be set to this value.
`postal_code`	array	no	If set, Postal Code will be set to this value.
`private_key_format`		no	

Parameter	Type	Required	Description
	string (, der, pem, pkcs8) (default: der)		Format for the returned private key. Generally the default will be controlled by the “format” parameter as either base64-encoded DER or PEM-encoded DER. However, this can be set to “pkcs8” to have the returned private key contain base64-encoded pkcs8 or PEM-encoded pkcs8 instead. Defaults to “der”.
`province`	array	no	If set, Province will be set to this value.
`serial_number`	string	no	The Subject’s requested serial number, if any. See RFC 4519 Section 2.31 ‘serialNumber’ for a description of this field. If you want more than one, specify alternative names in the alt_names map using OID 2.5.4.5. This has no impact on the final certificate’s Serial Number field.
`signature_bits`	integer (default: 0)	no	The number of bits to use in the signature algorithm; accepts 256 for SHA-2-256, 384 for SHA-2-384, and 512 for SHA-2-512. Defaults to 0 to automatically detect based on key length (SHA-2-256 for RSA keys, and matching the curve size for NIST P-Curves).
`street_address`	array	no	If set, Street Address will be set to this value.
`ttl`	integer	no	The requested Time To Live for the certificate; sets the expiration date. If not specified the role default, backend default, or

Parameter	Type	Required	Description
			system default TTL is used, in that order. Cannot be larger than the mount max TTL. Note: this only has an effect when generating a CA cert or signing a CA cert, not when generating a CSR for an intermediate CA.
`uri_sans`	array	no	The requested URI SANs, if any, in a comma-delimited list.

Responses

200: OK

Parameter	Type	Required	Description
`csr`	string	no	Certificate signing request.
`key_id`	string	no	Id of the key.
`private_key`	string	no	Generated private key.
`private_key_type`	string	no	Specifies the format used for marshaling the private key.

POST /{pki_mount_path}/intermediate/generate/{exported}

Operation ID: `pki-generate-intermediate`

Generate a new CSR and private key used for signing.

Parameters

Parameter	Type	Location	Required	Description
`exported`	string (internal, exported, kms)	path	yes	Must be “internal”, “exported” or “kms”. If set to “exported”, the generated private key will be returned. This is your *only* chance to retrieve the private key!
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`add_basic_constraints`	boolean	no	Whether to add a Basic Constraints extension with CA: true. Only needed as a workaround in some compatibility scenarios with Active Directory Certificate Services.
`alt_names`	string	no	The requested Subject Alternative Names, if any, in a comma-delimited list. May contain both DNS names and email addresses.
`common_name`	string	no	The requested common name; if you want more than one, specify the alternative names in the alt_names map. If not specified when signing, the common name will be taken from the CSR; other names must still be specified in alt_names or ip_sans.
`country`	array	no	If set, Country will be set to this value.
`exclude_cn_from_sans`	boolean (default: False)	no	If true, the Common Name will not be included in DNS or Email Subject Alternate Names. Defaults to false (CN is included).
`format`	string (pem, der, pem_bundle) (default: pem)	no	Format for returned data. Can be “pem”, “der”, or “pem_bundle”. If “pem_bundle”, any private key and issuing cert will be appended to the certificate pem. If “der”, the value will be base64 encoded. Defaults to “pem”.
`ip_sans`	array	no	The requested IP SANs, if any, in a comma-delimited list

Parameter	Type	Required	Description
`key_bits`	integer (default: 0)	no	The number of bits to use. Allowed values are 0 (universal default); with rsa key_type: 2048 (default), 3072, or 4096; with ec key_type: 224, 256 (default), 384, or 521; ignored with ed25519.
`key_name`	string	no	Provide a name to the generated or existing key, the name must be unique across all keys and not be the reserved value ‘default’
`key_ref`	string (default: default)	no	Reference to a existing key; either “default” for the configured default key, an identifier or the name assigned to the key.
`key_type`	string (rsa, ec, ed25519, gost3410-256-paramset-a, gost3410-256-paramset-b, gost3410-256-paramset-c, gost3410-256-paramset-d, gost3410-512-paramset-a, gost3410-512-paramset-b, gost3410-512-paramset-c) (default: rsa)	no	The type of key to use; defaults to RSA. “rsa” “ec”, “ed25519”, “gost3410-256-paramset-a”, “gost3410-256-paramset-b”, “gost3410-256-paramset-c”, “gost3410-256-paramset-d”, “gost3410-512-paramset-a”, “gost3410-512-paramset-b”, “gost3410-512-paramset-c” are the only valid values.
`locality`	array	no	If set, Locality will be set to this value.
`managed_key_id`	string	no	The name of the managed key to use when the exported type is

Parameter	Type	Required	Description
			kms. When kms type is the key type, this field or managed_key_name is required. Ignored for other types.
`managed_key_name`	string	no	The name of the managed key to use when the exported type is kms. When kms type is the key type, this field or managed_key_id is required. Ignored for other types.
`not_after`	string	no	Set the not after field of the certificate with specified date value. The value format should be given in UTC format YYYY-MM-ddTHH:MM:SSZ
`not_before_duration`	integer (default: 30)	no	The duration before now which the certificate needs to be backdated by.
`organization`	array	no	If set, O (Organization) will be set to this value.
`other_sans`	array	no	Requested other SANs, in an array with the format ;UTF8: for each entry.
`ou`	array	no	If set, OU (OrganizationalUnit) will be set to this value.
`postal_code`	array	no	If set, Postal Code will be set to this value.
`private_key_format`	string (, der, pem, pkcs8) (default: der)	no	Format for the returned private key. Generally the default will be controlled by the “format” parameter as either base64-encoded DER or PEM-encoded DER. However, this can be set to “pkcs8” to have the returned private key contain base64-

Parameter	Type	Required	Description
			encoded pkcs8 or PEM-encoded pkcs8 instead. Defaults to “der”.
`province`	array	no	If set, Province will be set to this value.
`serial_number`	string	no	The Subject’s requested serial number, if any. See RFC 4519 Section 2.31 ‘serialNumber’ for a description of this field. If you want more than one, specify alternative names in the alt_names map using OID 2.5.4.5. This has no impact on the final certificate’s Serial Number field.
`signature_bits`	integer (default: 0)	no	The number of bits to use in the signature algorithm; accepts 256 for SHA-2-256, 384 for SHA-2-384, and 512 for SHA-2-512. Defaults to 0 to automatically detect based on key length (SHA-2-256 for RSA keys, and matching the curve size for NIST P-Curves).
`street_address`	array	no	If set, Street Address will be set to this value.
`ttl`	integer	no	The requested Time To Live for the certificate; sets the expiration date. If not specified the role default, backend default, or system default TTL is used, in that order. Cannot be larger than the mount max TTL. Note: this only has an effect when generating a CA cert or signing a CA cert, not when generating a CSR for an intermediate CA.

Parameter	Type	Required	Description
`uri_sans`	array	no	The requested URI SANs, if any, in a comma-delimited list.

Responses

200: OK

Parameter	Type	Required	Description
`csr`	string	no	Certificate signing request.
`key_id`	string	no	Id of the key.
`private_key`	string	no	Generated private key.
`private_key_type`	string	no	Specifies the format used for marshaling the private key.

POST /{pki_mount_path}/intermediate/set-signed

Operation ID: `pki-set-signed-intermediate`

Provide the signed intermediate CA cert.

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`certificate`	string	no	PEM-format certificate. This must be a CA certificate with a public key matching the previously-generated key from the generation endpoint. Additional parent CAs may be optionally appended to the bundle.

Responses

200: OK

Parameter	Type	Required	Description
`existing_issuers`	array	no	Existing issuers specified as part of the import bundle of this request
`existing_keys`	array	no	Existing keys specified as part of the import bundle of this request
`imported_issuers`	array	no	Net-new issuers imported as a part of this request
`imported_keys`	array	no	Net-new keys imported as a part of this request
`mapping`	object	no	A mapping of issuer_id to key_id for all issuers included in this request

POST /{pki_mount_path}/issue/{role}

Operation ID: `pki-issue-with-role`

Request a certificate using a certain role with the provided details.

Parameters

Parameter	Type	Location	Required	Description
`role`	string	path	yes	The desired role with configuration for this request
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`alt_names`	string	no	The requested Subject Alternative Names, if any, in a comma-delimited list. If email protection is enabled for the role, this may contain email addresses.
`common_name`	string	no	The requested common name; if you want more than one, specify the alternative names in the alt_names map. If email protection is enabled in the role, this may be an email address.
`exclude_cn_from_sans`	boolean (default: False)	no	If true, the Common Name will not be included in DNS or Email Subject Alternate Names. Defaults to false (CN is included).
`format`	string (pem, der, pem_bundle) (default: pem)	no	Format for returned data. Can be “pem”, “der”, or “pem_bundle”. If “pem_bundle”, any private key and issuing cert will be appended to the certificate pem. If “der”, the value will be base64 encoded. Defaults to “pem”.
`ip_sans`	array	no	The requested IP SANs, if any, in a comma-delimited list
`issuer_ref`	string (default: default)	no	Reference to a existing issuer; either “default” for the configured default issuer, an identifier or the name assigned to the issuer.
`not_after`	string	no	Set the not after field of the certificate with specified date value. The value format should be given in UTC format YYYY-MM-ddTHH:MM:SSZ
`other_sans`	array	no	Requested other SANs, in an array with the format ;UTF8: for each entry.
`private_key_format`	string (, der, pem, pkcs8) (default: der)	no	Format for the returned private key. Generally the default will be controlled by the “format” parameter as either

Parameter	Type	Required	Description
			base64-encoded DER or PEM-encoded DER. However, this can be set to “pkcs8” to have the returned private key contain base64-encoded pkcs8 or PEM-encoded pkcs8 instead. Defaults to “der”.
`remove_roots_from_chain`	boolean (default: False)	no	Whether or not to remove self-signed CA certificates in the output of the ca_chain field.
`serial_number`	string	no	The Subject’s requested serial number, if any. See RFC 4519 Section 2.31 ‘serialNumber’ for a description of this field. If you want more than one, specify alternative names in the alt_names map using OID 2.5.4.5. This has no impact on the final certificate’s Serial Number field.
`ttl`	integer	no	The requested Time To Live for the certificate; sets the expiration date. If not specified the role default, backend default, or system default TTL is used, in that order. Cannot be larger than the role max TTL.
`uri_sans`	array	no	The requested URI SANs, if any, in a comma-delimited list.
`user_ids`	array	no	The requested user_ids value to place in the subject, if any, in a comma-delimited list. Restricted by allowed_user_ids. Any values are added with OID 0.9.2342.19200300.100.1.1.

Responses

200: OK

Parameter	Type	Required	Description
`ca_chain`	array	no	Certificate Chain
`certificate`	string	no	Certificate
`expiration`	integer	no	Time of expiration
`issuing_ca`	string	no	Issuing Certificate Authority
`private_key`	string	no	Private key
`private_key_type`	string	no	Private key type
`serial_number`	string	no	Serial Number

GET /{pki_mount_path}/issuer/{issuer_ref}

Operation ID: `pki-read-issuer`

Fetch a single issuer certificate.

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to a existing issuer; either “default” for the configured default issuer, an identifier or the name assigned to the issuer.
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`ca_chain`	array	no	CA Chain
`certificate`	string	no	Certificate
`crl_distribution_points`	array	no	CRL Distribution Points
`enable_aia_url_templating`	boolean	no	Whether or not templating is enabled for AIA fields
`issuer_id`	string	no	Issuer Id
`issuer_name`	string	no	Issuer Name
`issuing_certificates`	array	no	Issuing Certificates
`key_id`	string	no	Key Id
`leaf_not_after_behavior`	string	no	Leaf Not After Behavior
`manual_chain`	array	no	Manual Chain
`ocsp_servers`	array	no	OCSP Servers
`revocation_signature_algorithm`	string	no	Revocation Signature Alogrithm
`revocation_time`	integer	no	
`revocation_time_rfc3339`	string	no	
`revoked`	boolean	no	Revoked
`usage`	string	no	Usage

POST /{pki_mount_path}/issuer/{issuer_ref}

Operation ID: `pki-write-issuer`

Fetch a single issuer certificate.

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to a existing issuer; either “default” for the configured default issuer, an identifier or the name assigned to the issuer.
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`crl_distribution_points`	array	no	Comma-separated list of URLs to be used for the CRL distribution points attribute. See also RFC 5280 Section 4.2.1.13.
`enable_aia_url_templating`	boolean (default: False)	no	Whether or not to enabling templating of the above AIA fields. When templating is enabled the special values ‘{{issuer_id}}’, ‘{{cluster_path}}’, ‘{{cluster_aia_path}}’ are available, but the addresses are not checked for URL validity until issuance time. Using ‘{{cluster_path}}’ requires / config/cluster’s ‘path’ member to be set on all PR Secondary clusters and using ‘{{cluster_aia_path}}’ requires / config/cluster’s ‘aia_path’ member to be set on all PR secondary clusters.
`issuer_name`	string	no	Provide a name to the generated or existing issuer, the name must be unique across all issuers and not be the reserved value ‘default’
`issuing_certificates`	array	no	Comma-separated list of URLs to be used for the issuing certificate attribute. See also RFC 5280 Section 4.2.2.1.
`leaf_not_after_behavior`	string (default: err)	no	Behavior of leaf’s NotAfter fields: “err” to error if the computed NotAfter date exceeds that of this issuer; “truncate” to silently truncate to that of this issuer; or “permit” to allow this issuance to succeed

Parameter	Type	Required	Description
			(with NotAfter exceeding that of an issuer). Note that not all values will results in certificates that can be validated through the entire validity period. It is suggested to use “truncate” for intermediate CAs and “permit” only for root CAs.
`manual_chain`	array	no	Chain of issuer references to use to build this issuer’s computed CAChain field, when non-empty.
`ocsp_servers`	array	no	Comma-separated list of URLs to be used for the OCSP servers attribute. See also RFC 5280 Section 4.2.2.1.
`revocation_signature_algorithm`	string (default:)	no	Which x509.SignatureAlgorithm name to use for signing CRLs. This parameter allows differentiation between PKCS#1v1.5 and PSS keys and choice of signature hash algorithm. The default (empty string) value is for Go to select the signature algorithm. This can fail if the underlying key does not support the requested signature algorithm, which may not be known at modification time (such as with PKCS#11 managed RSA keys).
`usage`	array (default: [‘read-only’, ‘issuing-certificates’, ‘crl-signing’,	no	Comma-separated list (or string slice) of usages for this issuer; valid values are “read-only”, “issuing-certificates”, “crl-signing”, and “ocsp-signing”. Multiple values may be

Parameter	Type	Required	Description
	‘ocsp-signing’])		specified. Read-only is implicit and always set.

Responses

200: OK

Parameter	Type	Required	Description
`ca_chain`	array	no	CA Chain
`certificate`	string	no	Certificate
`crl_distribution_points`	array	no	CRL Distribution Points
`enable_aia_url_templating`	boolean	no	Whether or not templating is enabled for AIA fields
`issuer_id`	string	no	Issuer Id
`issuer_name`	string	no	Issuer Name
`issuing_certificates`	array	no	Issuing Certificates
`key_id`	string	no	Key Id
`leaf_not_after_behavior`	string	no	Leaf Not After Behavior
`manual_chain`	array	no	Manual Chain
`ocsp_servers`	array	no	OCSP Servers
`revocation_signature_algorithm`	string	no	Revocation Signature Alogrithm
`revocation_time`	integer	no	
`revocation_time_rfc3339`	string	no	
`revoked`	boolean	no	Revoked
`usage`	string	no	Usage

DELETE /{pki_mount_path}/issuer/{issuer_ref}

Operation ID: `pki-delete-issuer`

Fetch a single issuer certificate.

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to a existing issuer; either “default” for the configured default issuer, an identifier or the name assigned to the issuer.
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: No Content

POST /{pki_mount_path}/issuer/{issuer_ref}/acme/account/{kid}

Operation ID: `pki-write-issuer-issuer_ref-acme-account-kid`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to an existing issuer name or issuer id
`kid`	string	path	yes	The key identifier provided by the CA
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request ‘payload’ value
`protected`	string	no	ACME request ‘protected’ value
`signature`	string	no	ACME request ‘signature’ value

Responses

200: OK

POST /{pki_mount_path}/issuer/{issuer_ref}/acme/authorization/{auth_id}

Operation ID: `pki-write-issuer-issuer_ref-acme-authorization-auth_id`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`auth_id`	string	path	yes	ACME authorization identifier value
`issuer_ref`	string	path	yes	Reference to an existing issuer name or issuer id
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request ‘payload’ value
`protected`	string	no	ACME request ‘protected’ value
`signature`	string	no	ACME request ‘signature’ value

Responses

200: OK

POST /{pki_mount_path}/issuer/{issuer_ref}/acme/challenge/{auth_id}/{challenge_type}

Operation ID: `pki-write-issuer-issuer_ref-acme-challenge-auth_id-challenge_type`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`auth_id`	string	path	yes	ACME authorization identifier value
`challenge_type`	string	path	yes	ACME challenge type
`issuer_ref`	string	path	yes	Reference to an existing issuer name or issuer id
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request ‘payload’ value
`protected`	string	no	ACME request ‘protected’ value
`signature`	string	no	ACME request ‘signature’ value

Responses

200: OK

GET /{pki_mount_path}/issuer/{issuer_ref}/acme/directory

Operation ID: `pki-read-issuer-issuer_ref-acme-directory`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to an existing issuer name or issuer id
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /{pki_mount_path}/issuer/{issuer_ref}/acme/new-account

Operation ID: `pki-write-issuer-issuer_ref-acme-new-account`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to an existing issuer name or issuer id
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request ‘payload’ value
`protected`	string	no	ACME request ‘protected’ value
`signature`	string	no	ACME request ‘signature’ value

Responses

200: OK

POST /{pki_mount_path}/issuer/{issuer_ref}/acme/new-eab

Operation ID: `pki-generate-eab-key-for-issuer`

Generate external account bindings to be used for ACME

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to an existing issuer name or issuer id
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`acme_directory`	string	no	The ACME directory to which the key belongs
`created_on`	string	no	An RFC3339 formatted date time when the EAB token was created
`id`	string	no	The EAB key identifier
`key`	string	no	The EAB hmac key
`key_type`	string	no	The EAB key type

GET /{pki_mount_path}/issuer/{issuer_ref}/acme/new-nonce

Operation ID: `pki-read-issuer-issuer_ref-acme-new-nonce`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to an existing issuer name or issuer id
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /{pki_mount_path}/issuer/{issuer_ref}/acme/new-order

Operation ID: `pki-write-issuer-issuer_ref-acme-new-order`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to an existing issuer name or issuer id
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request ‘payload’ value
`protected`	string	no	ACME request ‘protected’ value
`signature`	string	no	ACME request ‘signature’ value

Responses

200: OK

POST /{pki_mount_path}/issuer/{issuer_ref}/acme/order/{order_id}

Operation ID: `pki-write-issuer-issuer_ref-acme-order-order_id`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to an existing issuer name or issuer id
`order_id`	string	path	yes	The ACME order identifier to fetch
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request 'payload' value
`protected`	string	no	ACME request 'protected' value
`signature`	string	no	ACME request 'signature' value

Responses

200: OK

POST /{pki_mount_path}/issuer/{issuer_ref}/acme/order/{order_id}/cert

Operation ID: `pki-write-issuer-issuer_ref-acme-order-order_id-cert`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to an existing issuer name or issuer id
`order_id`	string	path	yes	The ACME order identifier to fetch
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request ‘payload’ value
`protected`	string	no	ACME request ‘protected’ value
`signature`	string	no	ACME request ‘signature’ value

Responses

200: OK

POST /{pki_mount_path}/issuer/{issuer_ref}/acme/order/{order_id}/finalize

Operation ID: `pki-write-issuer-issuer_ref-acme-order-order_id-finalize`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to an existing issuer name or issuer id
`order_id`	string	path	yes	The ACME order identifier to fetch
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request ‘payload’ value
`protected`	string	no	ACME request ‘protected’ value
`signature`	string	no	ACME request ‘signature’ value

Responses

200: OK

POST /{pki_mount_path}/issuer/{issuer_ref}/acme/orders

Operation ID: `pki-write-issuer-issuer_ref-acme-orders`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to an existing issuer name or issuer id
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request ‘payload’ value
`protected`	string	no	ACME request ‘protected’ value
`signature`	string	no	ACME request ‘signature’ value

Responses

200: OK

POST /{pki_mount_path}/issuer/{issuer_ref}/acme/revoke-cert

Operation ID: `pki-write-issuer-issuer_ref-acme-revoke-cert`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to an existing issuer name or issuer id
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request ‘payload’ value
`protected`	string	no	ACME request ‘protected’ value
`signature`	string	no	ACME request ‘signature’ value

Responses

200: OK

GET /{pki_mount_path}/issuer/{issuer_ref}/crl

Operation ID: `pki-issuer-read-crl`

Fetch an issuer’s Certificate Revocation Log (CRL).

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to a existing issuer; either “default” for the configured default issuer, an identifier or the name assigned to the issuer.
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`crl`	string	no	

GET /{pki_mount_path}/issuer/{issuer_ref}/crl/delta

Operation ID: `pki-issuer-read-crl-delta`

Fetch an issuer’s Certificate Revocation Log (CRL).

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to a existing issuer; either “default” for the configured default issuer, an identifier or the name assigned to the issuer.
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`crl`	string	no	

GET /{pki_mount_path}/issuer/{issuer_ref}/crl/delta/der

Operation ID: `pki-issuer-read-crl-delta-der`

Fetch an issuer’s Certificate Revocation Log (CRL).

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to a existing issuer; either “default” for the configured default issuer, an identifier or the name assigned to the issuer.
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`crl`	string	no	

GET /{pki_mount_path}/issuer/{issuer_ref}/crl/delta/pem

Operation ID: `pki-issuer-read-crl-delta-pem`

Fetch an issuer’s Certificate Revocation Log (CRL).

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to a existing issuer; either “default” for the configured default issuer, an identifier or the name assigned to the issuer.
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`crl`	string	no	

GET /{pki_mount_path}/issuer/{issuer_ref}/crl/der

Operation ID: `pki-issuer-read-crl-der`

Fetch an issuer’s Certificate Revocation Log (CRL).

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to a existing issuer; either “default” for the configured default issuer, an identifier or the name assigned to the issuer.
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`crl`	string	no	

GET /{pki_mount_path}/issuer/{issuer_ref}/crl/pem

Operation ID: `pki-issuer-read-crl-pem`

Fetch an issuer’s Certificate Revocation Log (CRL).

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to a existing issuer; either “default” for the configured default issuer, an identifier or the name assigned to the issuer.
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`crl`	string	no	

GET /{pki_mount_path}/issuer/{issuer_ref}/der

Operation ID: `pki-read-issuer-der`

Fetch a single issuer certificate.

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to a existing issuer; either “default” for the configured default issuer, an identifier or the name assigned to the issuer.
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`ca_chain`	array	no	CA Chain
`certificate`	string	no	Certificate
`issuer_id`	string	no	Issuer Id
`issuer_name`	string	no	Issuer Name

304: Not Modified

POST /{pki_mount_path}/issuer/{issuer_ref}/issue/{role}

Operation ID: `pki-issuer-issue-with-role`

Request a certificate using a certain role with the provided details.

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to a existing issuer; either “default” for the configured default issuer, an identifier or the name assigned to the issuer.
`role`	string	path	yes	The desired role with configuration for this request
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`alt_names`	string	no	The requested Subject Alternative Names, if any, in a comma-delimited list. If email protection is enabled for the role, this may contain email addresses.
`common_name`	string	no	The requested common name; if you want more than one, specify the alternative names in the alt_names map. If email protection is enabled in the role, this may be an email address.
`exclude_cn_from_sans`	boolean (default: False)	no	If true, the Common Name will not be included in DNS or Email Subject Alternate Names. Defaults to false (CN is included).
`format`	string (pem, der, pem_bundle) (default: pem)	no	Format for returned data. Can be “pem”, “der”, or “pem_bundle”. If “pem_bundle”, any private key and issuing cert will be appended to the certificate pem. If “der”, the value will be base64 encoded. Defaults to “pem”.
`ip_sans`	array	no	The requested IP SANs, if any, in a comma-delimited list
`not_after`	string	no	Set the not after field of the certificate with specified date value. The value format should be given in UTC format YYYY-MM-ddTHH:MM:SSZ
`other_sans`	array	no	Requested other SANs, in an array with the format ;UTF8: for each entry.
`private_key_format`	string (, der, pem, pkcs8) (default: der)	no	Format for the returned private key. Generally the default will be controlled by the “format” parameter as either base64-encoded DER or PEM-encoded DER. However, this can be set to “pkcs8” to have the returned private key contain base64-encoded

Parameter	Type	Required	Description
			pkcs8 or PEM-encoded pkcs8 instead. Defaults to “der”.
`remove_roots_from_chain`	boolean (default: False)	no	Whether or not to remove self-signed CA certificates in the output of the ca_chain field.
`serial_number`	string	no	The Subject’s requested serial number, if any. See RFC 4519 Section 2.31 ‘serialNumber’ for a description of this field. If you want more than one, specify alternative names in the alt_names map using OID 2.5.4.5. This has no impact on the final certificate’s Serial Number field.
`ttl`	integer	no	The requested Time To Live for the certificate; sets the expiration date. If not specified the role default, backend default, or system default TTL is used, in that order. Cannot be larger than the role max TTL.
`uri_sans`	array	no	The requested URI SANs, if any, in a comma-delimited list.
`user_ids`	array	no	The requested user_ids value to place in the subject, if any, in a comma-delimited list. Restricted by allowed_user_ids. Any values are added with OID 0.9.2342.19200300.100.1.1.

Responses

200: OK

Parameter	Type	Required	Description
`ca_chain`	array	no	Certificate Chain
`certificate`	string	no	Certificate
`expiration`	integer	no	Time of expiration
`issuing_ca`	string	no	Issuing Certificate Authority
`private_key`	string	no	Private key
`private_key_type`	string	no	Private key type
`serial_number`	string	no	Serial Number

GET /{pki_mount_path}/issuer/{issuer_ref}/json

Operation ID: `pki-read-issuer-json`

Fetch a single issuer certificate.

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to a existing issuer; either “default” for the configured default issuer, an identifier or the name assigned to the issuer.
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`ca_chain`	array	no	CA Chain
`certificate`	string	no	Certificate
`issuer_id`	string	no	Issuer Id
`issuer_name`	string	no	Issuer Name

304: Not Modified

GET /{pki_mount_path}/issuer/{issuer_ref}/pem

Operation ID: `pki-read-issuer-pem`

Fetch a single issuer certificate.

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to a existing issuer; either “default” for the configured default issuer, an identifier or the name assigned to the issuer.
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`ca_chain`	array	no	CA Chain
`certificate`	string	no	Certificate
`issuer_id`	string	no	Issuer Id
`issuer_name`	string	no	Issuer Name

304: Not Modified

POST /{pki_mount_path}/issuer/{issuer_ref}/resign-crls

Operation ID: `pki-issuer-resign-crls`

Combine and sign with the provided issuer different CRLs

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to a existing issuer; either “default” for the configured default issuer, an identifier or the name assigned to the issuer.
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`crl_number`	integer	no	The sequence number to be written within the CRL Number extension.
`crls`	array	no	A list of PEM encoded CRLs to combine, originally signed by the requested issuer.
`delta_crl_base_number`	integer (default: -1)	no	Using a zero or greater value specifies the base CRL revision number to encode within a Delta CRL indicator extension, otherwise the extension will not be added.
`format`	string (default: pem)	no	The format of the combined CRL, can be “pem” or “der”. If “der”, the value will be base64 encoded. Defaults to “pem”.
`next_update`	string (default: 72h)	no	The amount of time the generated CRL should be valid; defaults to 72 hours.

Responses

200: OK

Parameter	Type	Required	Description
`crl`	string	no	CRL

POST /{pki_mount_path}/issuer/{issuer_ref}/revoke

Operation ID: `pki-revoke-issuer`

Revoke the specified issuer certificate.

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to a existing issuer; either “default” for the configured default issuer, an identifier or the name assigned to the issuer.
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`ca_chain`	array	no	Certificate Authority Chain
`certificate`	string	no	Certificate
`crl_distribution_points`	array	no	Specifies the URL values for the CRL Distribution Points field
`issuer_id`	string	no	ID of the issuer
`issuer_name`	string	no	Name of the issuer
`issuing_certificates`	array	no	Specifies the URL values for the Issuing Certificate field
`key_id`	string	no	ID of the Key
`leaf_not_after_behavior`	string	no	
`manual_chain`	array	no	Manual Chain
`ocsp_servers`	array	no	Specifies the URL values for the OCSP Servers field
`revocation_signature_algorithm`	string	no	Which signature algorithm to use when building CRLs
`revocation_time`	integer	no	Time of revocation
`revocation_time_rfc3339`	string	no	RFC formatted time of revocation
`revoked`	boolean	no	Whether the issuer was revoked
`usage`	string	no	Allowed usage

POST /{pki_mount_path}/issuer/{issuer_ref}/roles/{role}/acme/account/{kid}

Operation ID: `pki-write-issuer-issuer_ref-roles-role-acme-account-kid`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to an existing issuer name or issuer id
`kid`	string	path	yes	The key identifier provided by the CA
`role`	string	path	yes	The desired role for the acme request
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request ‘payload’ value
`protected`	string	no	ACME request ‘protected’ value
`signature`	string	no	ACME request ‘signature’ value

Responses

200: OK

POST /{pki_mount_path}/issuer/{issuer_ref}/roles/{role}/acme/authorization/{auth_id}

Operation ID: `pki-write-issuer-issuer_ref-roles-role-acme-authorization-auth_id`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`auth_id`	string	path	yes	ACME authorization identifier value
`issuer_ref`	string	path	yes	Reference to an existing issuer name or issuer id
`role`	string	path	yes	The desired role for the acme request
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request ‘payload’ value
`protected`	string	no	ACME request ‘protected’ value
`signature`	string	no	ACME request ‘signature’ value

Responses

200: OK

POST /{pki_mount_path}/issuer/{issuer_ref}/roles/{role}/acme/challenge/{auth_id}/{challenge_type}

Operation ID:
`pki-write-issuer-issuer_ref-roles-role-acme-challenge-auth_id-challenge_type`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`auth_id`	string	path	yes	ACME authorization identifier value
`challenge_type`	string	path	yes	ACME challenge type
`issuer_ref`	string	path	yes	Reference to an existing issuer name or issuer id
`role`	string	path	yes	The desired role for the acme request
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request ‘payload’ value
`protected`	string	no	ACME request ‘protected’ value
`signature`	string	no	ACME request ‘signature’ value

Responses

200: OK

GET /{pki_mount_path}/issuer/{issuer_ref}/roles/{role}/acme/directory

Operation ID: `pki-read-issuer-issuer_ref-roles-role-acme-directory`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to an existing issuer name or issuer id
`role`	string	path	yes	The desired role for the acme request
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /{pki_mount_path}/issuer/{issuer_ref}/roles/{role}/acme/new-account

Operation ID: `pki-write-issuer-issuer_ref-roles-role-acme-new-account`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to an existing issuer name or issuer id
`role`	string	path	yes	The desired role for the acme request
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request ‘payload’ value
`protected`	string	no	ACME request ‘protected’ value
`signature`	string	no	ACME request ‘signature’ value

Responses

200: OK

POST /{pki_mount_path}/issuer/{issuer_ref}/roles/{role}/acme/new-eab

Operation ID: `pki-generate-eab-key-for-issuer-and-role`

Generate external account bindings to be used for ACME

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to an existing issuer name or issuer id
`role`	string	path	yes	The desired role for the acme request
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`acme_directory`	string	no	The ACME directory to which the key belongs
`created_on`	string	no	An RFC3339 formatted date time when the EAB token was created
`id`	string	no	The EAB key identifier
`key`	string	no	The EAB hmac key
`key_type`	string	no	The EAB key type

GET /{pki_mount_path}/issuer/{issuer_ref}/roles/{role}/acme/new-nonce

Operation ID: `pki-read-issuer-issuer_ref-roles-role-acme-new-nonce`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to an existing issuer name or issuer id
`role`	string	path	yes	The desired role for the acme request
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /{pki_mount_path}/issuer/{issuer_ref}/roles/{role}/acme/new-order

Operation ID: `pki-write-issuer-issuer_ref-roles-role-acme-new-order`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to an existing issuer name or issuer id
`role`	string	path	yes	The desired role for the acme request
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request ‘payload’ value
`protected`	string	no	ACME request ‘protected’ value
`signature`	string	no	ACME request ‘signature’ value

Responses

200: OK

POST /{pki_mount_path}/issuer/{issuer_ref}/roles/{role}/acme/order/{order_id}

Operation ID: `pki-write-issuer-issuer_ref-roles-role-acme-order-order_id`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to an existing issuer name or issuer id
`order_id`	string	path	yes	The ACME order identifier to fetch
`role`	string	path	yes	The desired role for the acme request
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request ‘payload’ value
`protected`	string	no	ACME request ‘protected’ value
`signature`	string	no	ACME request ‘signature’ value

Responses

200: OK

POST /{pki_mount_path}/issuer/{issuer_ref}/roles/{role}/acme/order/{order_id}/cert

Operation ID: `pki-write-issuer-issuer_ref-roles-role-acme-order-order_id-cert`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to an existing issuer name or issuer id
`order_id`	string	path	yes	The ACME order identifier to fetch
`role`	string	path	yes	The desired role for the acme request
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request 'payload' value
`protected`	string	no	ACME request 'protected' value
`signature`	string	no	ACME request 'signature' value

Responses

200: OK

POST /{pki_mount_path}/issuer/{issuer_ref}/roles/{role}/acme/order/{order_id}/finalize

Operation ID: `pki-write-issuer-issuer_ref-roles-role-acme-order-order_id-finalize`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to an existing issuer name or issuer id
`order_id`	string	path	yes	The ACME order identifier to fetch
`role`	string	path	yes	The desired role for the acme request
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request ‘payload’ value
`protected`	string	no	ACME request ‘protected’ value
`signature`	string	no	ACME request ‘signature’ value

Responses

200: OK

POST /{pki_mount_path}/issuer/{issuer_ref}/roles/{role}/acme/orders

Operation ID: `pki-write-issuer-issuer_ref-roles-role-acme-orders`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to an existing issuer name or issuer id
`role`	string	path	yes	The desired role for the acme request
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request ‘payload’ value
`protected`	string	no	ACME request ‘protected’ value
`signature`	string	no	ACME request ‘signature’ value

Responses

200: OK

POST /{pki_mount_path}/issuer/{issuer_ref}/roles/{role}/acme/revoke-cert

Operation ID: `pki-write-issuer-issuer_ref-roles-role-acme-revoke-cert`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to an existing issuer name or issuer id
`role`	string	path	yes	The desired role for the acme request
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request ‘payload’ value
`protected`	string	no	ACME request ‘protected’ value
`signature`	string	no	ACME request ‘signature’ value

Responses

200: OK

POST /{pki_mount_path}/issuer/{issuer_ref}/sign-intermediate

Operation ID: `pki-issuer-sign-intermediate`

Issue an intermediate CA certificate based on the provided CSR.

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to a existing issuer; either “default” for the configured default issuer, an identifier or the name assigned to the issuer.
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`alt_names`	string	no	The requested Subject Alternative Names, if any, in a comma-delimited list. May contain both DNS names and email addresses.
`common_name`	string	no	The requested common name; if you want more than one, specify the alternative names in the alt_names map. If not specified when signing, the common name will be taken from the CSR; other names must still be specified in alt_names or ip_sans.
`country`	array	no	If set, Country will be set to this value.
`csr`	string (default:)	no	PEM-format CSR to be signed.
`exclude_cn_from_sans`	boolean (default: False)	no	If true, the Common Name will not be included in DNS or Email Subject Alternate Names. Defaults to false (CN is included).
`format`	string (pem, der, pem_bundle) (default: pem)	no	Format for returned data. Can be “pem”, “der”, or “pem_bundle”. If “pem_bundle”, any private key and issuing cert will be appended to the certificate pem. If “der”, the value will be base64 encoded. Defaults to “pem”.
`ip_sans`	array	no	The requested IP SANs, if any, in a comma-delimited list
`issuer_name`	string	no	Provide a name to the generated or existing issuer, the name must be unique across all issuers and not be the reserved value ‘default’
`locality`	array	no	If set, Locality will be set to this value.
`max_path_length`	integer (default: -1)	no	The maximum allowable path length
`not_after`	string	no	

Parameter	Type	Required	Description
			Set the not after field of the certificate with specified date value. The value format should be given in UTC format YYYY-MM-ddTHH:MM:SSZ
`not_before_duration`	integer (default: 30)	no	The duration before now which the certificate needs to be backdated by.
`organization`	array	no	If set, O (Organization) will be set to this value.
`other_sans`	array	no	Requested other SANs, in an array with the format ;UTF8: for each entry.
`ou`	array	no	If set, OU (OrganizationalUnit) will be set to this value.
`permitted_dns_domains`	array	no	Domains for which this certificate is allowed to sign or issue child certificates. If set, all DNS names (subject and alt) on child certs must be exact matches or subsets of the given domains (see https://tools.ietf.org/html/rfc5280#section-4.2.1.10).
`postal_code`	array	no	If set, Postal Code will be set to this value.
`private_key_format`	string (, der, pem, pkcs8) (default: der)	no	Format for the returned private key. Generally the default will be controlled by the “format” parameter as either base64-encoded DER or PEM-encoded DER. However, this can be set to “pkcs8” to have the returned private key contain base64-encoded pkcs8 or PEM-encoded pkcs8 instead. Defaults to “der”.
`province`	array	no	If set, Province will be set to this value.
`serial_number`	string	no	The Subject’s requested serial number, if any. See RFC 4519 Section 2.31 ‘serialNumber’ for a description of this field. If you want more than one, specify alternative names in the alt_names map

Parameter	Type	Required	Description
			using OID 2.5.4.5. This has no impact on the final certificate's Serial Number field.
`signature_bits`	integer (default: 0)	no	The number of bits to use in the signature algorithm; accepts 256 for SHA-2-256, 384 for SHA-2-384, and 512 for SHA-2-512. Defaults to 0 to automatically detect based on key length (SHA-2-256 for RSA keys, and matching the curve size for NIST P-Curves).
`skid`	string (default:)	no	Value for the Subject Key Identifier field (RFC 5280 Section 4.2.1.2). This value should ONLY be used when cross-signing to mimic the existing certificate's SKID value; this is necessary to allow certain TLS implementations (such as OpenSSL) which use SKID/AKID matches in chain building to restrict possible valid chains. Specified as a string in hex format. Default is empty, allowing Vault to automatically calculate the SKID according to method one in the above RFC section.
`street_address`	array	no	If set, Street Address will be set to this value.
`ttl`	integer	no	The requested Time To Live for the certificate; sets the expiration date. If not specified the role default, backend default, or system default TTL is used, in that order. Cannot be larger than the mount max TTL. Note: this only has an effect when generating a CA cert or signing a CA cert, not when generating a CSR for an intermediate CA.
`uri_sans`	array	no	The requested URI SANs, if any, in a comma-delimited list.
`use_csr_values`		no	If true, then: 1) Subject information, including names and alternate names,

Parameter	Type	Required	Description
	boolean (default: False)		will be preserved from the CSR rather than using values provided in the other parameters to this path; 2) Any key usages requested in the CSR will be added to the basic set of key usages used for CA certs signed by this path; for instance, the non-repudiation flag; 3) Extensions requested in the CSR will be copied into the issued certificate.
`use_pss`	boolean (default: False)	no	Whether or not to use PSS signatures when using a RSA key-type issuer. Defaults to false.

Responses

200: OK

Parameter	Type	Required	Description
`ca_chain`	array	no	CA Chain
`certificate`	string	no	Certificate
`expiration`	integer	no	Expiration Time
`issuing_ca`	string	no	Issuing CA
`serial_number`	string	no	Serial Number

POST /{pki_mount_path}/issuer/{issuer_ref}/sign-revocation-list

Operation ID: `pki-issuer-sign-revocation-list`

Generate and sign a CRL based on the provided parameters.

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to a existing issuer; either “default” for the configured default issuer, an identifier or the name assigned to the issuer.
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`crl_number`	integer	no	The sequence number to be written within the CRL Number extension.
`delta_crl_base_number`	integer (default: -1)	no	Using a zero or greater value specifies the base CRL revision number to encode within a Delta CRL indicator extension, otherwise the extension will not be added.
`extensions`	array	no	A list of maps containing extensions with keys id (string), critical (bool), value (string)
`format`	string (default: pem)	no	The format of the combined CRL, can be “pem” or “der”. If “der”, the value will be base64 encoded. Defaults to “pem”.
`next_update`	string (default: 72h)	no	The amount of time the generated CRL should be valid; defaults to 72 hours.
`revoked_certs`	array	no	A list of maps containing the keys serial_number (string), revocation_time (string), and extensions (map with keys id (string), critical (bool), value (string))

Responses

200: OK

Parameter	Type	Required	Description
`crl`	string	no	CRL

POST /{pki_mount_path}/issuer/{issuer_ref}/sign-self-issued

Operation ID: `pki-issuer-sign-self-issued`

Re-issue a self-signed certificate based on the provided certificate.

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to a existing issuer; either “default” for the configured default issuer, an identifier or the name assigned to the issuer.
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`certificate`	string	no	PEM-format self-issued certificate to be signed.
`require_matching_certificate_algorithms`	boolean (default: False)	no	If true, require the public key algorithm of the signer to match that of the self issued certificate.

Responses

200: OK

Parameter	Type	Required	Description
`certificate`	string	no	Certificate
`issuing_ca`	string	no	Issuing CA

POST /{pki_mount_path}/issuer/{issuer_ref}/sign-verbatim

Operation ID: `pki-issuer-sign-verbatim`

Issue a certificate directly based on the provided CSR.

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to a existing issuer; either “default” for the configured default issuer, an identifier or the name assigned to the issuer.
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`alt_names`	string	no	The requested Subject Alternative Names, if any, in a comma-delimited list. If email protection is enabled for the role, this may contain email addresses.
`common_name`	string	no	The requested common name; if you want more than one, specify the alternative names in the alt_names map. If email protection is enabled in the role, this may be an email address.
`csr`	string (default:)	no	PEM-format CSR to be signed. Values will be taken verbatim from the CSR, except for basic constraints.
`exclude_cn_from_sans`	boolean (default: False)	no	If true, the Common Name will not be included in DNS or Email Subject Alternate Names. Defaults to false (CN is included).
`ext_key_usage`	array (default: [])	no	A comma-separated string or list of extended key usages. Valid values can be found at https://golang.org/pkg/crypto/x509/#ExtKeyUsage – simply drop the “ExtKeyUsage” part of the name. To remove all key usages from being set, set this value to an empty list.
`ext_key_usage_oids`	array	no	A comma-separated string or list of extended key usage oids.
`format`	string (pem, der, pem_bundle) (default: pem)	no	Format for returned data. Can be “pem”, “der”, or “pem_bundle”. If “pem_bundle”, any private key and issuing cert

Parameter	Type	Required	Description
			will be appended to the certificate pem. If “der”, the value will be base64 encoded. Defaults to “pem”.
`ip_sans`	array	no	The requested IP SANs, if any, in a comma-delimited list
`key_usage`	array (default: [‘DigitalSignature’, ‘KeyAgreement’, ‘KeyEncipherment’])	no	A comma-separated string or list of key usages (not extended key usages). Valid values can be found at https://golang.org/pkg/crypto/x509/#KeyUsage – simply drop the “KeyUsage” part of the name. To remove all key usages from being set, set this value to an empty list.
`not_after`	string	no	Set the not after field of the certificate with specified date value. The value format should be given in UTC format YYYY-MM-ddTHH:MM:SSZ
`other_sans`	array	no	Requested other SANs, in an array with the format ;UTF8: for each entry.
`private_key_format`	string (, der, pem, pkcs8) (default: der)	no	Format for the returned private key. Generally the default will be controlled by the “format” parameter as either base64-encoded DER or PEM-encoded DER. However, this can be set to “pkcs8” to have the returned private key contain base64-encoded pkcs8 or PEM-encoded pkcs8 instead. Defaults to “der”.
`remove_roots_from_chain`	boolean (default: False)	no	

Parameter	Type	Required	Description
			Whether or not to remove self-signed CA certificates in the output of the ca_chain field.
`role`	string	no	The desired role with configuration for this request
`serial_number`	string	no	The Subject's requested serial number, if any. See RFC 4519 Section 2.31 'serialNumber' for a description of this field. If you want more than one, specify alternative names in the alt_names map using OID 2.5.4.5. This has no impact on the final certificate's Serial Number field.
`signature_bits`	integer (default: 0)	no	The number of bits to use in the signature algorithm; accepts 256 for SHA-2-256, 384 for SHA-2-384, and 512 for SHA-2-512. Defaults to 0 to automatically detect based on key length (SHA-2-256 for RSA keys, and matching the curve size for NIST P-Curves).
`ttl`	integer	no	The requested Time To Live for the certificate; sets the expiration date. If not specified the role default, backend default, or system default TTL is used, in that order. Cannot be larger than the role max TTL.
`uri_sans`	array	no	The requested URI SANs, if any, in a comma-delimited list.
`use_pss`	boolean (default: False)	no	Whether or not to use PSS signatures when using a RSA

Parameter	Type	Required	Description
			key-type issuer. Defaults to false.
`user_ids`	array	no	The requested user_ids value to place in the subject, if any, in a comma-delimited list. Restricted by allowed_user_ids. Any values are added with OID 0.9.2342.19200300.100.1.1.

Responses

200: OK

Parameter	Type	Required	Description
`ca_chain`	array	no	Certificate Chain
`certificate`	string	no	Certificate
`expiration`	integer	no	Time of expiration
`issuing_ca`	string	no	Issuing Certificate Authority
`private_key`	string	no	Private key
`private_key_type`	string	no	Private key type
`serial_number`	string	no	Serial Number

POST /{pki_mount_path}/issuer/{issuer_ref}/sign-verbatim/{role}

Operation ID: `pki-issuer-sign-verbatim-with-role`

Issue a certificate directly based on the provided CSR.

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to a existing issuer; either “default” for the configured default issuer, an identifier or the name assigned to the issuer.
`role`	string	path	yes	The desired role with configuration for this request
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`alt_names`	string	no	The requested Subject Alternative Names, if any, in a comma-delimited list. If email protection is enabled for the role, this may contain email addresses.
`common_name`	string	no	The requested common name; if you want more than one, specify the alternative names in the alt_names map. If email protection is enabled in the role, this may be an email address.
`csr`	string (default:)	no	PEM-format CSR to be signed. Values will be taken verbatim from the CSR, except for basic constraints.
`exclude_cn_from_sans`	boolean (default: False)	no	If true, the Common Name will not be included in DNS or Email Subject Alternate Names. Defaults to false (CN is included).
`ext_key_usage`	array (default: [])	no	A comma-separated string or list of extended key usages. Valid values can be found at https://golang.org/pkg/crypto/x509/#ExtKeyUsage – simply drop the “ExtKeyUsage” part of the name. To remove all key usages from being set, set this value to an empty list.
`ext_key_usage_oids`	array	no	A comma-separated string or list of extended key usage oids.
`format`	string (pem, der, pem_bundle) (default: pem)	no	Format for returned data. Can be “pem”, “der”, or “pem_bundle”. If “pem_bundle”, any private key and issuing cert

Parameter	Type	Required	Description
			will be appended to the certificate pem. If “der”, the value will be base64 encoded. Defaults to “pem”.
`ip_sans`	array	no	The requested IP SANs, if any, in a comma-delimited list
`key_usage`	array (default: [‘DigitalSignature’, ‘KeyAgreement’, ‘KeyEncipherment’])	no	A comma-separated string or list of key usages (not extended key usages). Valid values can be found at https://golang.org/pkg/crypto/x509/#KeyUsage – simply drop the “KeyUsage” part of the name. To remove all key usages from being set, set this value to an empty list.
`not_after`	string	no	Set the not after field of the certificate with specified date value. The value format should be given in UTC format YYYY-MM-ddTHH:MM:SSZ
`other_sans`	array	no	Requested other SANs, in an array with the format ;UTF8: for each entry.
`private_key_format`	string (, der, pem, pkcs8) (default: der)	no	Format for the returned private key. Generally the default will be controlled by the “format” parameter as either base64-encoded DER or PEM-encoded DER. However, this can be set to “pkcs8” to have the returned private key contain base64-encoded pkcs8 or PEM-encoded pkcs8 instead. Defaults to “der”.
`remove_roots_from_chain`	boolean (default: False)	no	

Parameter	Type	Required	Description
			Whether or not to remove self-signed CA certificates in the output of the ca_chain field.
`serial_number`	string	no	The Subject's requested serial number, if any. See RFC 4519 Section 2.31 'serialNumber' for a description of this field. If you want more than one, specify alternative names in the alt_names map using OID 2.5.4.5. This has no impact on the final certificate's Serial Number field.
`signature_bits`	integer (default: 0)	no	The number of bits to use in the signature algorithm; accepts 256 for SHA-2-256, 384 for SHA-2-384, and 512 for SHA-2-512. Defaults to 0 to automatically detect based on key length (SHA-2-256 for RSA keys, and matching the curve size for NIST P-Curves).
`ttl`	integer	no	The requested Time To Live for the certificate; sets the expiration date. If not specified the role default, backend default, or system default TTL is used, in that order. Cannot be larger than the role max TTL.
`uri_sans`	array	no	The requested URI SANs, if any, in a comma-delimited list.
`use_pss`	boolean (default: False)	no	Whether or not to use PSS signatures when using a RSA key-type issuer. Defaults to false.
`user_ids`	array	no	The requested user_ids value to place in the subject, if any, in a

Parameter	Type	Required	Description
			comma-delimited list. Restricted by allowed_user_ids. Any values are added with OID 0.9.2342.19200300.100.1.1.

Responses

200: OK

Parameter	Type	Required	Description
`ca_chain`	array	no	Certificate Chain
`certificate`	string	no	Certificate
`expiration`	integer	no	Time of expiration
`issuing_ca`	string	no	Issuing Certificate Authority
`private_key`	string	no	Private key
`private_key_type`	string	no	Private key type
`serial_number`	string	no	Serial Number

POST /{pki_mount_path}/issuer/{issuer_ref}/sign/{role}

Operation ID: `pki-issuer-sign-with-role`

Request certificates using a certain role with the provided details.

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to a existing issuer; either “default” for the configured default issuer, an identifier or the name assigned to the issuer.
`role`	string	path	yes	The desired role with configuration for this request
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`alt_names`	string	no	The requested Subject Alternative Names, if any, in a comma-delimited list. If email protection is enabled for the role, this may contain email addresses.
`common_name`	string	no	The requested common name; if you want more than one, specify the alternative names in the alt_names map. If email protection is enabled in the role, this may be an email address.
`csr`	string (default:)	no	PEM-format CSR to be signed.
`exclude_cn_from_sans`	boolean (default: False)	no	If true, the Common Name will not be included in DNS or Email Subject Alternate Names. Defaults to false (CN is included).
`format`	string (pem, der, pem_bundle) (default: pem)	no	Format for returned data. Can be “pem”, “der”, or “pem_bundle”. If “pem_bundle”, any private key and issuing cert will be appended to the certificate pem. If “der”, the value will be base64 encoded. Defaults to “pem”.
`ip_sans`	array	no	The requested IP SANs, if any, in a comma-delimited list
`not_after`	string	no	Set the not after field of the certificate with specified date value. The value format should be given in UTC format YYYY-MM-ddTHH:MM:SSZ
`other_sans`	array	no	Requested other SANs, in an array with the format ;UTF8: for each entry.
`private_key_format`	string (, der, pem, pkcs8) (default: der)	no	Format for the returned private key. Generally the default will be controlled by the “format” parameter as either base64-encoded DER or PEM-encoded DER. However, this can be

Parameter	Type	Required	Description
			set to “pkcs8” to have the returned private key contain base64-encoded pkcs8 or PEM-encoded pkcs8 instead. Defaults to “der”.
`remove_roots_from_chain`	boolean (default: False)	no	Whether or not to remove self-signed CA certificates in the output of the ca_chain field.
`serial_number`	string	no	The Subject’s requested serial number, if any. See RFC 4519 Section 2.31 ‘serialNumber’ for a description of this field. If you want more than one, specify alternative names in the alt_names map using OID 2.5.4.5. This has no impact on the final certificate’s Serial Number field.
`ttl`	integer	no	The requested Time To Live for the certificate; sets the expiration date. If not specified the role default, backend default, or system default TTL is used, in that order. Cannot be larger than the role max TTL.
`uri_sans`	array	no	The requested URI SANs, if any, in a comma-delimited list.
`user_ids`	array	no	The requested user_ids value to place in the subject, if any, in a comma-delimited list. Restricted by allowed_user_ids. Any values are added with OID 0.9.2342.19200300.100.1.1.

Responses

200: OK

Parameter	Type	Required	Description
`ca_chain`	array	no	Certificate Chain
`certificate`	string	no	Certificate
`expiration`	integer	no	Time of expiration
`issuing_ca`	string	no	Issuing Certificate Authority
`private_key`	string	no	Private key
`private_key_type`	string	no	Private key type
`serial_number`	string	no	Serial Number

GET /{pki_mount_path}/issuer/{issuer_ref}/unified-crl

Operation ID: `pki-issuer-read-unified-crl`

Fetch an issuer's Certificate Revocation Log (CRL).

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to a existing issuer; either "default" for the configured default issuer, an identifier or the name assigned to the issuer.
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`crl`	string	no	

GET /{pki_mount_path}/issuer/{issuer_ref}/unified-crl/delta

Operation ID: `pki-issuer-read-unified-crl-delta`

Fetch an issuer's Certificate Revocation Log (CRL).

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to a existing issuer; either “default” for the configured default issuer, an identifier or the name assigned to the issuer.
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`crl`	string	no	

GET /{pki_mount_path}/issuer/{issuer_ref}/unified-crl/delta/der

Operation ID: `pki-issuer-read-unified-crl-delta-der`

Fetch an issuer’s Certificate Revocation Log (CRL).

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to a existing issuer; either “default” for the configured default issuer, an identifier or the name assigned to the issuer.
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`crl`	string	no	

GET /{pki_mount_path}/issuer/{issuer_ref}/unified-crl/delta/pem

Operation ID: `pki-issuer-read-unified-crl-delta-pem`

Fetch an issuer’s Certificate Revocation Log (CRL).

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to a existing issuer; either “default” for the configured default issuer, an identifier or the name assigned to the issuer.
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`crl`	string	no	

GET /{pki_mount_path}/issuer/{issuer_ref}/unified-crl/der

Operation ID: `pki-issuer-read-unified-crl-der`

Fetch an issuer’s Certificate Revocation Log (CRL).

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to a existing issuer; either “default” for the configured default issuer, an identifier or the name assigned to the issuer.
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`crl`	string	no	

GET /{pki_mount_path}/issuer/{issuer_ref}/unified-crl/pem

Operation ID: `pki-issuer-read-unified-crl-pem`

Fetch an issuer’s Certificate Revocation Log (CRL).

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`issuer_ref`	string	path	yes	Reference to a existing issuer; either “default” for the configured default issuer, an identifier or the name assigned to the issuer.
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`crl`	string	no	

GET /{pki_mount_path}/issuers

Operation ID: `pki-list-issuers`

Fetch a list of CA certificates.

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

Parameter	Type	Required	Description
`key_info`	object	no	Key info with issuer name
`keys`	array	no	A list of keys

POST /{pki_mount_path}/issuers/generate/intermediate/{exported}

Operation ID: `pki-issuers-generate-intermediate`

Generate a new CSR and private key used for signing.

Parameters

Parameter	Type	Location	Required	Description
`exported`	string (internal, exported, kms)	path	yes	Must be “internal”, “exported” or “kms”. If set to “exported”, the generated private key will be returned. This is your *only* chance to retrieve the private key!
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`add_basic_constraints`	boolean	no	Whether to add a Basic Constraints extension with CA: true. Only needed as a workaround in some compatibility scenarios with Active Directory Certificate Services.
`alt_names`	string	no	The requested Subject Alternative Names, if any, in a comma-delimited list. May contain both DNS names and email addresses.
`common_name`	string	no	The requested common name; if you want more than one, specify the alternative names in the alt_names map. If not specified when signing, the common name will be taken from the CSR; other names must still be specified in alt_names or ip_sans.
`country`	array	no	If set, Country will be set to this value.
`exclude_cn_from_sans`	boolean (default: False)	no	If true, the Common Name will not be included in DNS or Email Subject Alternate Names. Defaults to false (CN is included).
`format`	string (pem, der, pem_bundle) (default: pem)	no	Format for returned data. Can be “pem”, “der”, or “pem_bundle”. If “pem_bundle”, any private key and issuing cert will be appended to the certificate pem. If “der”, the value will be base64 encoded. Defaults to “pem”.
`ip_sans`	array	no	The requested IP SANs, if any, in a comma-delimited list

Parameter	Type	Required	Description
`key_bits`	integer (default: 0)	no	The number of bits to use. Allowed values are 0 (universal default); with rsa key_type: 2048 (default), 3072, or 4096; with ec key_type: 224, 256 (default), 384, or 521; ignored with ed25519.
`key_name`	string	no	Provide a name to the generated or existing key, the name must be unique across all keys and not be the reserved value ‘default’
`key_ref`	string (default: default)	no	Reference to a existing key; either “default” for the configured default key, an identifier or the name assigned to the key.
`key_type`	string (rsa, ec, ed25519, gost3410-256-paramset-a, gost3410-256-paramset-b, gost3410-256-paramset-c, gost3410-256-paramset-d, gost3410-512-paramset-a, gost3410-512-paramset-b, gost3410-512-paramset-c) (default: rsa)	no	The type of key to use; defaults to RSA. “rsa” “ec”, “ed25519”, “gost3410-256-paramset-a”, “gost3410-256-paramset-b”, “gost3410-256-paramset-c”, “gost3410-256-paramset-d”, “gost3410-512-paramset-a”, “gost3410-512-paramset-b”, “gost3410-512-paramset-c” are the only valid values.
`locality`	array	no	If set, Locality will be set to this value.
`managed_key_id`	string	no	The name of the managed key to use when the exported type is

Parameter	Type	Required	Description
			kms. When kms type is the key type, this field or managed_key_name is required. Ignored for other types.
`managed_key_name`	string	no	The name of the managed key to use when the exported type is kms. When kms type is the key type, this field or managed_key_id is required. Ignored for other types.
`not_after`	string	no	Set the not after field of the certificate with specified date value. The value format should be given in UTC format YYYY-MM-ddTHH:MM:SSZ
`not_before_duration`	integer (default: 30)	no	The duration before now which the certificate needs to be backdated by.
`organization`	array	no	If set, O (Organization) will be set to this value.
`other_sans`	array	no	Requested other SANs, in an array with the format ;UTF8: for each entry.
`ou`	array	no	If set, OU (OrganizationalUnit) will be set to this value.
`postal_code`	array	no	If set, Postal Code will be set to this value.
`private_key_format`	string (, der, pem, pkcs8) (default: der)	no	Format for the returned private key. Generally the default will be controlled by the “format” parameter as either base64-encoded DER or PEM-encoded DER. However, this can be set to “pkcs8” to have the returned private key contain base64-

Parameter	Type	Required	Description
			encoded pkcs8 or PEM-encoded pkcs8 instead. Defaults to “der”.
`province`	array	no	If set, Province will be set to this value.
`serial_number`	string	no	The Subject’s requested serial number, if any. See RFC 4519 Section 2.31 ‘serialNumber’ for a description of this field. If you want more than one, specify alternative names in the alt_names map using OID 2.5.4.5. This has no impact on the final certificate’s Serial Number field.
`signature_bits`	integer (default: 0)	no	The number of bits to use in the signature algorithm; accepts 256 for SHA-2-256, 384 for SHA-2-384, and 512 for SHA-2-512. Defaults to 0 to automatically detect based on key length (SHA-2-256 for RSA keys, and matching the curve size for NIST P-Curves).
`street_address`	array	no	If set, Street Address will be set to this value.
`ttl`	integer	no	The requested Time To Live for the certificate; sets the expiration date. If not specified the role default, backend default, or system default TTL is used, in that order. Cannot be larger than the mount max TTL. Note: this only has an effect when generating a CA cert or signing a CA cert, not when generating a CSR for an intermediate CA.

Parameter	Type	Required	Description
`uri_sans`	array	no	The requested URI SANs, if any, in a comma-delimited list.

Responses

200: OK

Parameter	Type	Required	Description
`csr`	string	no	Certificate signing request.
`key_id`	string	no	Id of the key.
`private_key`	string	no	Generated private key.
`private_key_type`	string	no	Specifies the format used for marshaling the private key.

POST /{pki_mount_path}/issuers/generate/root/{exported}

Operation ID: `pki-issuers-generate-root`

Generate a new CA certificate and private key used for signing.

Parameters

| Parameter | Type | Location | Required | Description |

| ———- | ——— | ———— | ———— | ———- | | `exported` | string (internal, exported, kms) | path | yes | Must be “internal”, “exported” or “kms”. If set to “exported”, the generated private key will be returned. This is your *only* chance to retrieve the private key! | | `pki_mount_path` | string | path | yes | Path that the backend was mounted at |

Request body parameters

Parameter	Type	Required	Description
`alt_names`	string	no	The requested Subject Alternative Names, if any, in a comma-delimited list. May contain both DNS names and email addresses.
`common_name`	string	no	The requested common name; if you want more than one, specify the alternative names in the alt_names map. If not specified when signing, the common name will be taken from the CSR; other names must still be specified in alt_names or ip_sans.
`country`	array	no	If set, Country will be set to this value.
`exclude_cn_from_sans`	boolean (default: False)	no	If true, the Common Name will not be included in DNS or Email Subject Alternate Names. Defaults to false (CN is included).
`format`	string (pem, der, pem_bundle) (default: pem)	no	Format for returned data. Can be “pem”, “der”, or “pem_bundle”. If “pem_bundle”, any private key and issuing cert will be appended to the certificate pem. If “der”, the value will be base64 encoded. Defaults to “pem”.
`ip_sans`	array	no	The requested IP SANs, if any, in a comma-delimited list
`issuer_name`	string	no	Provide a name to the generated or existing issuer, the name must be unique across all issuers and not be the reserved value ‘default’
`key_bits`	integer (default: 0)	no	The number of bits to use. Allowed values are 0 (universal default); with rsa key_type: 2048 (default), 3072, or 4096; with ec key_type:

Parameter	Type	Required	Description
			224, 256 (default), 384, or 521; ignored with ed25519.
`key_name`	string	no	Provide a name to the generated or existing key, the name must be unique across all keys and not be the reserved value ‘default’
`key_ref`	string (default: default)	no	Reference to a existing key; either “default” for the configured default key, an identifier or the name assigned to the key.
`key_type`	string (rsa, ec, ed25519, gost3410-256-paramset-a, gost3410-256-paramset-b, gost3410-256-paramset-c, gost3410-256-paramset-d, gost3410-512-paramset-a, gost3410-512-paramset-b, gost3410-512-paramset-c) (default: rsa)	no	The type of key to use; defaults to RSA. “rsa” “ec”, “ed25519”, “gost3410-256-paramset-a”, “gost3410-256-paramset-b”, “gost3410-256-paramset-c”, “gost3410-256-paramset-d”, “gost3410-512-paramset-a”, “gost3410-512-paramset-b”, “gost3410-512-paramset-c” are the only valid values.
`locality`	array	no	If set, Locality will be set to this value.
`managed_key_id`	string	no	The name of the managed key to use when the exported type is kms. When kms type is the key type, this field or managed_key_name is required. Ignored for other types.
`managed_key_name`	string	no	The name of the managed key to use when the exported type is

Parameter	Type	Required	Description
			kms. When kms type is the key type, this field or managed_key_id is required. Ignored for other types.
`max_path_length`	integer (default: -1)	no	The maximum allowable path length
`not_after`	string	no	Set the not after field of the certificate with specified date value. The value format should be given in UTC format YYYY-MM-ddTHH:MM:SSZ
`not_before_duration`	integer (default: 30)	no	The duration before now which the certificate needs to be backdated by.
`organization`	array	no	If set, O (Organization) will be set to this value.
`other_sans`	array	no	Requested other SANs, in an array with the format ;UTF8: for each entry.
`ou`	array	no	If set, OU (OrganizationalUnit) will be set to this value.
`permitted_dns_domains`	array	no	Domains for which this certificate is allowed to sign or issue child certificates. If set, all DNS names (subject and alt) on child certs must be exact matches or subsets of the given domains (see https://tools.ietf.org/html/rfc5280#section-4.2.1.10).
`postal_code`	array	no	If set, Postal Code will be set to this value.
`private_key_format`	string (, der, pem, pkcs8) (default: der)	no	Format for the returned private key. Generally the default will be controlled by the “format” parameter as either base64-encoded DER or PEM-encoded

Parameter	Type	Required	Description
			DER. However, this can be set to “pkcs8” to have the returned private key contain base64-encoded pkcs8 or PEM-encoded pkcs8 instead. Defaults to “der”.
`province`	array	no	If set, Province will be set to this value.
`serial_number`	string	no	The Subject’s requested serial number, if any. See RFC 4519 Section 2.31 ‘serialNumber’ for a description of this field. If you want more than one, specify alternative names in the alt_names map using OID 2.5.4.5. This has no impact on the final certificate’s Serial Number field.
`signature_bits`	integer (default: 0)	no	The number of bits to use in the signature algorithm; accepts 256 for SHA-2-256, 384 for SHA-2-384, and 512 for SHA-2-512. Defaults to 0 to automatically detect based on key length (SHA-2-256 for RSA keys, and matching the curve size for NIST P-Curves).
`street_address`	array	no	If set, Street Address will be set to this value.
`ttl`	integer	no	The requested Time To Live for the certificate; sets the expiration date. If not specified the role default, backend default, or system default TTL is used, in that order. Cannot be larger than the mount max TTL. Note: this only has an effect when generating a CA cert or signing a CA cert, not when generating a CSR for an intermediate CA.

Parameter	Type	Required	Description
`uri_sans`	array	no	The requested URI SANs, if any, in a comma-delimited list.
`use_pss`	boolean (default: False)	no	Whether or not to use PSS signatures when using a RSA key-type issuer. Defaults to false.

Responses

200: OK

Parameter	Type	Required	Description
`certificate`	string	no	The generated self-signed CA certificate.
`expiration`	integer	no	The expiration of the given issuer.
`issuer_id`	string	no	The ID of the issuer
`issuer_name`	string	no	The name of the issuer.
`issuing_ca`	string	no	The issuing certificate authority.
`key_id`	string	no	The ID of the key.
`key_name`	string	no	The key name if given.
`private_key`	string	no	The private key if exported was specified.
`serial_number`	string	no	The requested Subject's named serial number.

POST /{pki_mount_path}/issuers/import/bundle

Operation ID: `pki-issuers-import-bundle`

Import the specified issuing certificates.

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`pem_bundle`	string	no	PEM-format, concatenated unencrypted secret-key (optional) and certificates.

Responses

200: OK

Parameter	Type	Required	Description
`existing_issuers`	array	no	Existing issuers specified as part of the import bundle of this request
`existing_keys`	array	no	Existing keys specified as part of the import bundle of this request
`imported_issuers`	array	no	Net-new issuers imported as a part of this request
`imported_keys`	array	no	Net-new keys imported as a part of this request
`mapping`	object	no	A mapping of issuer_id to key_id for all issuers included in this request

POST /{pki_mount_path}/issuers/import/cert

Operation ID: `pki-issuers-import-cert`

Import the specified issuing certificates.

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`pem_bundle`	string	no	PEM-format, concatenated unencrypted secret-key (optional) and certificates.

Responses

200: OK

Parameter	Type	Required	Description
`existing_issuers`	array	no	Existing issuers specified as part of the import bundle of this request
`existing_keys`	array	no	Existing keys specified as part of the import bundle of this request
`imported_issuers`	array	no	Net-new issuers imported as a part of this request
`imported_keys`	array	no	Net-new keys imported as a part of this request
`mapping`	object	no	A mapping of issuer_id to key_id for all issuers included in this request

GET /{pki_mount_path}/key/{key_ref}

Operation ID: `pki-read-key`

Fetch a single issuer key

Parameters

Parameter	Type	Location	Required	Description
`key_ref`	string	path	yes	Reference to key; either “default” for the configured default key, an identifier of a key, or the name assigned to the key.
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`key_id`	string	no	Key Id
`key_name`	string	no	Key Name
`key_type`	string	no	Key Type
`managed_key_id`	string	no	Managed Key Id
`managed_key_name`	string	no	Managed Key Name
`subject_key_id`	string	no	RFC 5280 Subject Key Identifier of the public counterpart

POST /{pki_mount_path}/key/{key_ref}

Operation ID: `pki-write-key`

Fetch a single issuer key

Parameters

Parameter	Type	Location	Required	Description
`key_ref`	string	path	yes	Reference to key; either “default” for the configured default key, an identifier of a key, or the name assigned to the key.
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`key_name`	string	no	Human-readable name for this key.

Responses

204: OK

Parameter	Type	Required	Description
`key_id`	string	no	Key Id
`key_name`	string	no	Key Name
`key_type`	string	no	Key Type

DELETE /{pki_mount_path}/key/{key_ref}

Operation ID: `pki-delete-key`

Fetch a single issuer key

Parameters

Parameter	Type	Location	Required	Description
`key_ref`	string	path	yes	Reference to key; either “default” for the configured default key, an identifier of a key, or the name assigned to the key.
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: No Content

GET /{pki_mount_path}/keys

Operation ID: `pki-list-keys`

Fetch a list of all issuer keys

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

Parameter	Type	Required	Description
`key_info`	object	no	Key info with issuer name
`keys`	array	no	A list of keys

POST /{pki_mount_path}/keys/generate/exported

Operation ID: `pki-generate-exported-key`

Generate a new private key used for signing.

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`key_bits`	integer (default: 0)	no	The number of bits to use. Allowed values are 0 (universal default); with rsa key_type: 2048 (default), 3072, or 4096; with ec key_type: 224, 256 (default), 384, or 521; ignored with ed25519.
`key_name`	string	no	Optional name to be used for this key
`key_type`	string (rsa, ec, ed25519, gost3410-256-paramset-a, gost3410-256-paramset-b, gost3410-256-paramset-c, gost3410-256-paramset-d, gost3410-512-paramset-a, gost3410-512-paramset-b, gost3410-512-paramset-c) (default: rsa)	no	The type of key to use; defaults to RSA. “rsa” “ec”, “ed25519”, “gost3410-256-paramset-a”, “gost3410-256-paramset-b”, “gost3410-256-paramset-c”, “gost3410-256-paramset-d”, “gost3410-512-paramset-a”, “gost3410-512-paramset-b”, “gost3410-512-paramset-c” are the only valid values.
`managed_key_id`	string	no	The name of the managed key to use when the exported type is kms. When kms type is the key type, this field or managed_key_name is required. Ignored for other types.
`managed_key_name`	string	no	The name of the managed key to use when the exported type is kms. When kms type is the key type, this field or managed_key_id is required. Ignored for other types.

Responses

200: OK

Parameter	Type	Required	Description
`key_id`	string	no	ID assigned to this key.
`key_name`	string	no	Name assigned to this key.
`key_type`	string	no	The type of key to use; defaults to RSA. “rsa” “ec”, “ed25519”, “gost3410-256-paramset-a”, “gost3410-256-paramset-b”, “gost3410-256-paramset-c”, “gost3410-256-paramset-d”, “gost3410-512-paramset-a”, “gost3410-512-paramset-b”, “gost3410-512-paramset-c” are the only valid values.
`private_key`	string	no	The private key string

POST /{pki_mount_path}/keys/generate/internal

Operation ID: `pki-generate-internal-key`

Generate a new private key used for signing.

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`key_bits`	integer (default: 0)	no	The number of bits to use. Allowed values are 0 (universal default); with rsa key_type: 2048 (default), 3072, or 4096; with ec key_type: 224, 256 (default), 384, or 521; ignored with ed25519.
`key_name`	string	no	Optional name to be used for this key
`key_type`	string (rsa, ec, ed25519, gost3410-256-paramset-a, gost3410-256-paramset-b, gost3410-256-paramset-c, gost3410-256-paramset-d, gost3410-512-paramset-a, gost3410-512-paramset-b, gost3410-512-paramset-c) (default: rsa)	no	The type of key to use; defaults to RSA. “rsa” “ec”, “ed25519”, “gost3410-256-paramset-a”, “gost3410-256-paramset-b”, “gost3410-256-paramset-c”, “gost3410-256-paramset-d”, “gost3410-512-paramset-a”, “gost3410-512-paramset-b”, “gost3410-512-paramset-c” are the only valid values.
`managed_key_id`	string	no	The name of the managed key to use when the exported type is kms. When kms type is the key type, this field or managed_key_name is required. Ignored for other types.
`managed_key_name`	string	no	The name of the managed key to use when the exported type is kms. When kms type is the key type, this field or managed_key_id is required. Ignored for other types.

Responses

200: OK

Parameter	Type	Required	Description
`key_id`	string	no	ID assigned to this key.
`key_name`	string	no	Name assigned to this key.
`key_type`	string	no	The type of key to use; defaults to RSA. “rsa” “ec”, “ed25519”, “gost3410-256-paramset-a”, “gost3410-256-paramset-b”, “gost3410-256-paramset-c”, “gost3410-256-paramset-d”, “gost3410-512-paramset-a”, “gost3410-512-paramset-b”, “gost3410-512-paramset-c” are the only valid values.
`private_key`	string	no	The private key string

POST /{pki_mount_path}/keys/generate/kms

Operation ID: `pki-generate-kms-key`

Generate a new private key used for signing.

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`key_bits`	integer (default: 0)	no	The number of bits to use. Allowed values are 0 (universal default); with rsa key_type: 2048 (default), 3072, or 4096; with ec key_type: 224, 256 (default), 384, or 521; ignored with ed25519.
`key_name`	string	no	Optional name to be used for this key
`key_type`	string (rsa, ec, ed25519, gost3410-256-paramset-a, gost3410-256-paramset-b, gost3410-256-paramset-c, gost3410-256-paramset-d, gost3410-512-paramset-a, gost3410-512-paramset-b, gost3410-512-paramset-c) (default: rsa)	no	The type of key to use; defaults to RSA. “rsa” “ec”, “ed25519”, “gost3410-256-paramset-a”, “gost3410-256-paramset-b”, “gost3410-256-paramset-c”, “gost3410-256-paramset-d”, “gost3410-512-paramset-a”, “gost3410-512-paramset-b”, “gost3410-512-paramset-c” are the only valid values.
`managed_key_id`	string	no	The name of the managed key to use when the exported type is kms. When kms type is the key type, this field or managed_key_name is required. Ignored for other types.
`managed_key_name`	string	no	The name of the managed key to use when the exported type is kms. When kms type is the key type, this field or managed_key_id is required. Ignored for other types.

Responses

200: OK

Parameter	Type	Required	Description
`key_id`	string	no	ID assigned to this key.
`key_name`	string	no	Name assigned to this key.
`key_type`	string	no	The type of key to use; defaults to RSA. “rsa” “ec”, “ed25519”, “gost3410-256-paramset-a”, “gost3410-256-paramset-b”, “gost3410-256-paramset-c”, “gost3410-256-paramset-d”, “gost3410-512-paramset-a”, “gost3410-512-paramset-b”, “gost3410-512-paramset-c” are the only valid values.
`private_key`	string	no	The private key string

POST /{pki_mount_path}/keys/import

Operation ID: `pki-import-key`

Import the specified key.

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`key_name`	string	no	Optional name to be used for this key
`pem_bundle`	string	no	PEM-format, unencrypted secret key

Responses

200: OK

Parameter	Type	Required	Description
`key_id`	string	no	ID assigned to this key.
`key_name`	string	no	Name assigned to this key.
`key_type`	string	no	The type of key to use; defaults to RSA. “rsa” “ec”, “ed25519”, “gost3410-256-paramset-a”, “gost3410-256-paramset-b”, “gost3410-256-paramset-c”, “gost3410-256-paramset-d”, “gost3410-512-paramset-a”, “gost3410-512-paramset-b”, “gost3410-512-paramset-c” are the only valid values.

POST /{pki_mount_path}/ocsp

Operation ID: `pki-query-ocsp`

Query a certificate’s revocation status through OCSP’

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

GET /{pki_mount_path}/ocsp/{req}

Operation ID: `pki-query-ocsp-with-get-req`

Query a certificate’s revocation status through OCSP’

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`req`	string	path	yes	base-64 encoded ocsp request
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /{pki_mount_path}/revoke

Operation ID: `pki-revoke`

Revoke a certificate by serial number or with explicit certificate. When calling /revoke-with-key, the private key corresponding to the certificate must be provided to authenticate the request.

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`certificate`	string	no	Certificate to revoke in PEM format; must be signed by an issuer in this mount.
`serial_number`	string	no	Certificate serial number, in colon- or hyphen-separated octal

Responses

200: OK

Parameter	Type	Required	Description
`revocation_time`	integer	no	Revocation Time
`revocation_time_rfc3339`	string	no	Revocation Time
`state`	string	no	Revocation State

POST /{pki_mount_path}/revoke-with-key

Operation ID: `pki-revoke-with-key`

Revoke a certificate by serial number or with explicit certificate. When calling /revoke-with-key, the private key corresponding to the certificate must be provided to authenticate the request.

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`certificate`	string	no	Certificate to revoke in PEM format; must be signed by an issuer in this mount.
`private_key`	string	no	Key to use to verify revocation permission; must be in PEM format.
`serial_number`	string	no	Certificate serial number, in colon- or hyphen-separated octal

Responses

200: OK

Parameter	Type	Required	Description
`revocation_time`	integer	no	Revocation Time
`revocation_time_rfc3339`	string	no	Revocation Time
`state`	string	no	Revocation State

GET /{pki_mount_path}/roles

Operation ID: `pki-list-roles`

List the existing roles in this backend

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

Parameter	Type	Required	Description
`keys`	array	no	List of roles

GET /{pki_mount_path}/roles/{name}

Operation ID: `pki-read-role`

Manage the roles that can be created with this backend.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the role
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`allow_any_name`	boolean	no	If set, clients can request certificates for any domain, regardless of allowed_domains restrictions. See the documentation for more information.
`allow_bare_domains`	boolean	no	If set, clients can request certificates for the base domains themselves, e.g. “example.com” of domains listed in allowed_domains. This is a separate option as in some cases this can be considered a security threat. See the documentation for more information.
`allow_glob_domains`	boolean	no	If set, domains specified in allowed_domains can include shell-style glob patterns, e.g. “ftp*.example.com”. See the documentation for more information.
`allow_ip_sans`	boolean	no	If set, IP Subject Alternative Names are allowed. Any valid IP is accepted and No authorization checking is performed.
`allow_localhost`	boolean	no	Whether to allow “localhost” and “localdomain” as a valid common name in a request, independent of allowed_domains value.
`allow_subdomains`	boolean	no	If set, clients can request certificates for subdomains of domains listed in allowed_domains, including wildcard subdomains. See the documentation for more information.

Parameter	Type	Required	Description
`allow_token_displayname`	boolean	no	Whether to allow “localhost” and “localdomain” as a valid common name in a request, independent of allowed_domains value.
`allow_wildcard_certificates`	boolean	no	If set, allows certificates with wildcards in the common name to be issued, conforming to RFC 6125’s Section 6.4.3; e.g., “*.example.net” or “b*z.example.net”. See the documentation for more information.
`allowed_domains`	array	no	Specifies the domains this role is allowed to issue certificates for. This is used with the allow_bare_domains, allow_subdomains, and allow_glob_domains to determine matches for the common name, DNS-typed SAN entries, and Email-typed SAN entries of certificates. See the documentation for more information. This parameter accepts a comma-separated string or list of domains.
`allowed_domains_template`	boolean	no	If set, Allowed domains can be specified using identity template policies. Non-templated domains are also permitted.
`allowed_other_sans`	array	no	If set, an array of allowed other names to put in SANs. These values support globbing and must be in the format ;:. Currently only “utf8” is a valid type. All values, including globbing values, must use this syntax, with the exception being a single “*” which

Parameter	Type	Required	Description
			allows any OID and any value (but type must still be utf8).
`allowed_serial_numbers`	array	no	If set, an array of allowed serial numbers to put in Subject. These values support globbing.
`allowed_uri_sans`	array	no	If set, an array of allowed URIs for URI Subject Alternative Names. Any valid URI is accepted, these values support globbing.
`allowed_uri_sans_template`	boolean	no	If set, Allowed URI SANs can be specified using identity template policies. Non-templated URI SANs are also permitted.
`allowed_user_ids`	array	no	If set, an array of allowed user-ids to put in user system login name specified here: https://www.rfc-editor.org/rfc/rfc1274#section-9.3.1
`basic_constraints_valid_for_non_ca`	boolean	no	Mark Basic Constraints valid when issuing non-CA certificates.
`client_flag`	boolean	no	If set, certificates are flagged for client auth use. Defaults to true. See also RFC 5280 Section 4.2.1.12.
`cn_validations`	array	no	List of allowed validations to run against the Common Name field. Values can include 'email' to validate the CN is a email address, 'hostname' to validate the CN is a valid hostname (potentially including wildcards). When multiple validations are specified, these take OR semantics (either email OR hostname are allowed). The

Parameter	Type	Required	Description
			special value ‘disabled’ allows disabling all CN name validations, allowing for arbitrary non-Hostname, non-Email address CNs.
`code_signing_flag`	boolean	no	If set, certificates are flagged for code signing use. Defaults to false. See also RFC 5280 Section 4.2.1.12.
`country`	array	no	If set, Country will be set to this value in certificates issued by this role.
`email_protection_flag`	boolean	no	If set, certificates are flagged for email protection use. Defaults to false. See also RFC 5280 Section 4.2.1.12.
`enforce_hostnames`	boolean	no	If set, only valid host names are allowed for CN and DNS SANs, and the host part of email addresses. Defaults to true.
`ext_key_usage`	array	no	A comma-separated string or list of extended key usages. Valid values can be found at https://golang.org/pkg/crypto/x509/#ExtKeyUsage – simply drop the “ExtKeyUsage” part of the name. To remove all key usages from being set, set this value to an empty list. See also RFC 5280 Section 4.2.1.12.
`ext_key_usage_oids`	array	no	A comma-separated string or list of extended key usage oids.
`generate_lease`	boolean	no	If set, certificates issued/signed against this role will have Vault leases attached to them. Defaults to “false”. Certificates can be

Parameter	Type	Required	Description
			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.
`issuer_ref`	string	no	Reference to the issuer used to sign requests serviced by this role.
`key_bits`	integer	no	The number of bits to use. Allowed values are 0 (universal default); with rsa key_type: 2048 (default), 3072, or 4096; with ec key_type: 224, 256 (default), 384, or 521; ignored with ed25519.
`key_type`	string	no	The type of key to use; defaults to RSA. “rsa” “ec”, “ed25519”, “gost3410-256-paramset-a”, “gost3410-256-paramset-b”, “gost3410-256-paramset-c”, “gost3410-256-paramset-d”, “gost3410-512-paramset-a”, “gost3410-512-paramset-b”, “gost3410-512-paramset-c” and “any” are the only valid values.
`key_usage`	array	no	A comma-separated string or list of key usages (not extended key usages). Valid values can be found at https://golang.org/pkg/

Parameter	Type	Required	Description
			crypto/x509/#KeyUsage – simply drop the “KeyUsage” part of the name. To remove all key usages from being set, set this value to an empty list. See also RFC 5280 Section 4.2.1.3.
`locality`	array	no	If set, Locality will be set to this value in certificates issued by this role.
`max_ttl`	integer	no	The maximum allowed lease duration. If not set, defaults to the system maximum lease TTL.
`no_store`	boolean	no	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”.
`not_after`	string	no	Set the not after field of the certificate with specified date value. The value format should be given in UTC format YYYY-MM-ddTHH:MM:SSZ.
`not_before_duration`	integer	no	The duration in seconds before now which the certificate needs to be backdated by.
`organization`	array	no	If set, O (Organization) will be set to this value in certificates issued by this role.

Parameter	Type	Required	Description
`ou`	array	no	If set, OU (OrganizationalUnit) will be set to this value in certificates issued by this role.
`policy_identifiers`	array	no	A comma-separated string or list of policy OIDs, or a JSON list of qualified policy information, which must include an oid, and may include a notice and/or cps url, using the form [{“oid”=“1.3.6.1.4.1.7.8”,“notice”=“I am a user Notice”}, {“oid”=“1.3.6.1.4.1.44947.1.2.4 “,“cps”=“https://example.com”}].
`postal_code`	array	no	If set, Postal Code will be set to this value in certificates issued by this role.
`province`	array	no	If set, Province will be set to this value in certificates issued by this role.
`require_cn`	boolean	no	If set to false, makes the ‘common_name’ field optional while generating a certificate.
`server_flag`	boolean (default: True)	no	If set, certificates are flagged for server auth use. Defaults to true. See also RFC 5280 Section 4.2.1.12.
`signature_bits`	integer	no	The number of bits to use in the signature algorithm; accepts 256 for SHA-2-256, 384 for SHA-2-384, and 512 for SHA-2-512. Defaults to 0 to automatically detect based on key length (SHA-2-256 for RSA keys, and matching the curve size for NIST P-Curves).
`single_valued_rdns`		no	

Parameter	Type	Required	Description
	boolean (default: False)		If set to true, each subject attribute value (Country, Organization, OU, etc.) is encoded as its own separate single-valued RDN in the certificate. This matches the common practice of most CAs (OpenSSL, Microsoft CA). When false, all values of the same attribute type are grouped into a single multi-valued RDN.
`street_address`	array	no	If set, Street Address will be set to this value in certificates issued by this role.
`ttl`	integer	no	The lease duration (validity period of the certificate) if no specific lease duration is requested. The lease duration controls the expiration of certificates issued by this backend. Defaults to the system default value or the value of max_ttl, whichever is shorter.
`use_csr_common_name`	boolean	no	If set, when used with a signing profile, the common name in the CSR will be used. This does *not* include any requested Subject Alternative Names; use use_csr_sans for that. Defaults to true.
`use_csr_sans`	boolean	no	If set, when used with a signing profile, the SANs in the CSR will be used. This does *not* include the Common Name (cn); use use_csr_common_name for that. Defaults to true.
`use_pss`	boolean	no	

Parameter	Type	Required	Description
			Whether or not to use PSS signatures when using a RSA key-type issuer. Defaults to false.

POST /{pki_mount_path}/roles/{name}

Operation ID: `pki-write-role`

Manage the roles that can be created with this backend.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the role
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`allow_any_name`	boolean	no	If set, clients can request certificates for any domain, regardless of allowed_domains restrictions. See the documentation for more information.
`allow_bare_domains`	boolean	no	If set, clients can request certificates for the base domain themselves, e.g. “example.com of domains listed in allowed_domains. This is a separate option as in some cas this can be considered a securi threat. See the documentation f more information.
`allow_glob_domains`	boolean	no	If set, domains specified in allowed_domains can include shell-style glob patterns, e.g. “ftp*.example.com”. See the documentation for more information.
`allow_ip_sans`	boolean (default: True)	no	If set, IP Subject Alternative Names are allowed. Any valid I is accepted and No authorizatio checking is performed.
`allow_localhost`	boolean (default: True)	no	Whether to allow “localhost” an “localdomain” as a valid commo name in a request, independen of allowed_domains value.
`allow_subdomains`	boolean	no	If set, clients can request certificates for subdomains of domains listed in allowed_domains, including wildcard subdomains. See the documentation for more information.

Parameter	Type	Required	Description
`allow_wildcard_certificates`	boolean (default: True)	no	If set, allows certificates with wildcards in the common name be issued, conforming to RFC 6125's Section 6.4.3; e.g., "*.example.net" or "*bz*.example.net". See the documentation for more information.
`allowed_domains`	array	no	Specifies the domains this role allowed to issue certificates for. This is used with the allow_bare_domains, allow_subdomains, and allow_glob_domains to determi matches for the common name DNS-typed SAN entries, and Email-typed SAN entries of certificates. See the documentation for more information. This parameter accepts a comma-separated string or list of domains.
`allowed_domains_template`	boolean (default: False)	no	If set, Allowed domains can be specified using identity templat policies. Non-templated domain are also permitted.
`allowed_other_sans`	array	no	If set, an array of allowed other names to put in SANs. These values support globbing and mu be in the format ;:. Currently on "utf8" is a valid type. All values, including globbing values, must use this syntax, with the exception being a single "*" whi allows any OID and any value (but type must still be utf8).
`allowed_serial_numbers`	array	no	

Parameter	Type	Required	Description
			If set, an array of allowed serial numbers to put in Subject. Thes values support globbing.
`allowed_uri_sans`	array	no	If set, an array of allowed URIs for URI Subject Alternative Names. Any valid URI is accepted, these values support globbing.
`allowed_uri_sans_template`	boolean (default: False)	no	If set, Allowed URI SANs can b specified using identity templat policies. Non-templated URI SANs are also permitted.
`allowed_user_ids`	array	no	If set, an array of allowed user- to put in user system login nam specified here: https://www.rfc-editor.org/rfc/rfc1274#section-9.3.1
`backend`	string	no	Backend Type
`basic_constraints_valid_for_non_ca`	boolean	no	Mark Basic Constraints valid when issuing non-CA certificate
`client_flag`	boolean (default: True)	no	If set, certificates are flagged fo client auth use. Defaults to true See also RFC 5280 Section 4.2.1.12.
`cn_validations`	array (default: [’email’, ‘hostname’])	no	List of allowed validations to ru against the Common Name fiel Values can include ’email’ to validate the CN is a email address, ‘hostname’ to validate the CN is a valid hostname (potentially including wildcards) When multiple validations are specified, these take OR semantics (either email OR hostname are allowed). The special value ‘disabled’ allows

Parameter	Type	Required	Description
			disabling all CN name validatio allowing for arbitrary non-Hostname, non-Email address CNs.
`code_signing_flag`	boolean	no	If set, certificates are flagged fo code signing use. Defaults to false. See also RFC 5280 Secti 4.2.1.12.
`country`	array	no	If set, Country will be set to this value in certificates issued by th role.
`email_protection_flag`	boolean	no	If set, certificates are flagged fo email protection use. Defaults t false. See also RFC 5280 Secti 4.2.1.12.
`enforce_hostnames`	boolean (default: True)	no	If set, only valid host names are allowed for CN and DNS SANs and the host part of email addresses. Defaults to true.
`ext_key_usage`	array (default: [])	no	A comma-separated string or lis of extended key usages. Valid values can be found at https://golang.org/pkg/crypto/x509/#ExtKeyUsage – simply drop th "ExtKeyUsage" part of the nam To remove all key usages from being set, set this value to an empty list. See also RFC 5280 Section 4.2.1.12.
`ext_key_usage_oids`	array	no	A comma-separated string or lis of extended key usage oids.
`generate_lease`	boolean	no	If set, certificates issued/signed against this role will have Vault leases attached to them. Defau to "false". Certificates can be added to the CRL by "vault

Parameter	Type	Required	Description
			revoke <lease_id>” when certificates are associated with leases. It can also be done usin the “pki/revoke” endpoint. However, when lease generatio is disabled, invoking “pki/revoke would be the only way to add th certificates to the CRL. When large number of certificates are generated with long lifetimes, it recommended that lease generation be disabled, as large amount of leases adversely affe the startup time of Vault.
`issuer_ref`	string (default: default)	no	Reference to the issuer used to sign requests serviced by this role.
`key_bits`	integer (default: 0)	no	The number of bits to use. Allowed values are 0 (universal default); with rsa key_type: 204 (default), 3072, or 4096; with ec key_type: 224, 256 (default), 38 or 521; ignored with ed25519.
`key_type`	string (rsa, ec, ed25519, gost3410-256-paramset-a, gost3410-256-paramset-b, gost3410-256-paramset-c, gost3410-256-paramset-d, gost3410-512-paramset-a, gost3410-512-paramset-b, gost3410-512-	no	The type of key to use; defaults RSA. “rsa” “ec”, “ed25519”, “gost3410-256-paramset-a”, “gost3410-256-paramset-b”, “gost3410-256-paramset-c”, “gost3410-256-paramset-d”, “gost3410-512-paramset-a”, “gost3410-512-paramset-b”, “gost3410-512-paramset-c” and “any” are the only valid values.

Parameter	Type	Required	Description
	paramset-c, any) (default: rsa)		
`key_usage`	array (default: [‘DigitalSignature’, ‘KeyAgreement’, ‘KeyEncipherment’])	no	A comma-separated string or lis of key usages (not extended ke usages). Valid values can be found at https://golang.org/pkg/ crypto/x509/#KeyUsage – simp drop the “KeyUsage” part of the name. To remove all key usage from being set, set this value to an empty list. See also RFC 52 Section 4.2.1.3.
`locality`	array	no	If set, Locality will be set to this value in certificates issued by th role.
`max_ttl`	integer	no	The maximum allowed lease duration. If not set, defaults to t system maximum lease TTL.
`no_store`	boolean	no	If set, certificates issued/signed against this role will not be stor in the storage backend. This ca improve performance when issuing large numbers of certificates. However, certificate issued in this way cannot be enumerated or revoked, so this option is recommended only for certificates that are non-sensitiv or extremely short-lived. This option implies a value of “false” for “generate_lease”.
`not_after`	string	no	Set the not after field of the certificate with specified date value. The value format should given in UTC format YYYY-MM ddTHH:MM:SSZ.

Parameter	Type	Required	Description
`not_before_duration`	integer (default: 30)	no	The duration before now which the certificate needs to be backdated by.
`organization`	array	no	If set, O (Organization) will be s to this value in certificates issue by this role.
`ou`	array	no	If set, OU (OrganizationalUnit) be set to this value in certificate issued by this role.
`policy_identifiers`	array	no	A comma-separated string or lis of policy OIDs, or a JSON list o qualified policy information, whi must include an oid, and may include a notice and/or cps url, using the form [{“oid”=“1.3.6.1.4.1.7.8”,“notice” am a user Notice”}, {“oid”=“1.3.6.1.4.1.44947.1.2.4 “,“cps”=“https://example.com”}].
`postal_code`	array	no	If set, Postal Code will be set to this value in certificates issued this role.
`province`	array	no	If set, Province will be set to thi value in certificates issued by th role.
`require_cn`	boolean (default: True)	no	If set to false, makes the ‘common_name’ field optional while generating a certificate.
`server_flag`	boolean (default: True)	no	If set, certificates are flagged fo server auth use. Defaults to tru See also RFC 5280 Section 4.2.1.12.
`signature_bits`	integer (default: 0)	no	The number of bits to use in the signature algorithm; accepts 25 for SHA-2-256, 384 for

Parameter	Type	Required	Description
			SHA-2-384, and 512 for SHA-2-512. Defaults to 0 to automatically detect based on key length (SHA-2-256 for RSA keys, and matching the curve s for NIST P-Curves).
`single_valued_rdns`	boolean (default: False)	no	If set to true, each subject attribute value (Country, Organization, OU, etc.) is encoded as its own separate single-valued RDN in the certificate. This matches the common practice of most CAs (OpenSSL, Microsoft CA). Whe false, all values of the same attribute type are grouped into a single multi-valued RDN.
`street_address`	array	no	If set, Street Address will be se this value in certificates issued this role.
`ttl`	integer	no	The lease duration (validity peri of the certificate) if no specific lease duration is requested. Th lease duration controls the expiration of certificates issued this backend. Defaults to the system default value or the valu of max_ttl, whichever is shorter
`use_csr_common_name`	boolean (default: True)	no	If set, when used with a signing profile, the common name in th CSR will be used. This does *n include any requested Subject Alternative Names; use use_csr_sans for that. Defaults true.
`use_csr_sans`	boolean (default: True)	no	If set, when used with a signing profile, the SANs in the CSR wi

Parameter	Type	Required	Description
			be used. This does *not* includ the Common Name (cn); use use_csr_common_name for tha Defaults to true.
`use_pss`	boolean (default: False)	no	Whether or not to use PSS signatures when using a RSA key-type issuer. Defaults to fals

Responses

200: OK

Parameter	Type	Required	Description
`allow_any_name`	boolean	no	If set, clients can request certificates for any domain, regardless of allowed_domains restrictions. See the documentation for more information.
`allow_bare_domains`	boolean	no	If set, clients can request certificates for the base domains themselves, e.g. “example.com” of domains listed in allowed_domains. This is a separate option as in some cases this can be considered a security threat. See the documentation for more information.
`allow_glob_domains`	boolean	no	If set, domains specified in allowed_domains can include shell-style glob patterns, e.g. “ftp*.example.com”. See the documentation for more information.
`allow_ip_sans`	boolean	no	If set, IP Subject Alternative Names are allowed. Any valid IP is accepted and No authorization checking is performed.
`allow_localhost`	boolean	no	Whether to allow “localhost” and “localdomain” as a valid common name in a request, independent of allowed_domains value.
`allow_subdomains`	boolean	no	If set, clients can request certificates for subdomains of domains listed in allowed_domains, including wildcard subdomains. See the documentation for more information.

Parameter	Type	Required	Description
`allow_token_displayname`	boolean	no	Whether to allow “localhost” and “localdomain” as a valid common name in a request, independent of allowed_domains value.
`allow_wildcard_certificates`	boolean	no	If set, allows certificates with wildcards in the common name to be issued, conforming to RFC 6125’s Section 6.4.3; e.g., “*.example.net” or “b*z.example.net”. See the documentation for more information.
`allowed_domains`	array	no	Specifies the domains this role is allowed to issue certificates for. This is used with the allow_bare_domains, allow_subdomains, and allow_glob_domains to determine matches for the common name, DNS-typed SAN entries, and Email-typed SAN entries of certificates. See the documentation for more information. This parameter accepts a comma-separated string or list of domains.
`allowed_domains_template`	boolean	no	If set, Allowed domains can be specified using identity template policies. Non-templated domains are also permitted.
`allowed_other_sans`	array	no	If set, an array of allowed other names to put in SANs. These values support globbing and must be in the format ;:. Currently only “utf8” is a valid type. All values, including globbing values, must use this syntax, with the exception being a single “*” which

Parameter	Type	Required	Description
			allows any OID and any value (but type must still be utf8).
`allowed_serial_numbers`	array	no	If set, an array of allowed serial numbers to put in Subject. These values support globbing.
`allowed_uri_sans`	array	no	If set, an array of allowed URIs for URI Subject Alternative Names. Any valid URI is accepted, these values support globbing.
`allowed_uri_sans_template`	boolean	no	If set, Allowed URI SANs can be specified using identity template policies. Non-templated URI SANs are also permitted.
`allowed_user_ids`	array	no	If set, an array of allowed user-ids to put in user system login name specified here: https://www.rfc-editor.org/rfc/rfc1274#section-9.3.1
`basic_constraints_valid_for_non_ca`	boolean	no	Mark Basic Constraints valid when issuing non-CA certificates.
`client_flag`	boolean	no	If set, certificates are flagged for client auth use. Defaults to true. See also RFC 5280 Section 4.2.1.12.
`cn_validations`	array	no	List of allowed validations to run against the Common Name field. Values can include 'email' to validate the CN is a email address, 'hostname' to validate the CN is a valid hostname (potentially including wildcards). When multiple validations are specified, these take OR semantics (either email OR hostname are allowed). The

Parameter	Type	Required	Description
			special value ‘disabled’ allows disabling all CN name validations, allowing for arbitrary non-Hostname, non-Email address CNs.
`code_signing_flag`	boolean	no	If set, certificates are flagged for code signing use. Defaults to false. See also RFC 5280 Section 4.2.1.12.
`country`	array	no	If set, Country will be set to this value in certificates issued by this role.
`email_protection_flag`	boolean	no	If set, certificates are flagged for email protection use. Defaults to false. See also RFC 5280 Section 4.2.1.12.
`enforce_hostnames`	boolean	no	If set, only valid host names are allowed for CN and DNS SANs, and the host part of email addresses. Defaults to true.
`ext_key_usage`	array	no	A comma-separated string or list of extended key usages. Valid values can be found at https://golang.org/pkg/crypto/x509/#ExtKeyUsage – simply drop the “ExtKeyUsage” part of the name. To remove all key usages from being set, set this value to an empty list. See also RFC 5280 Section 4.2.1.12.
`ext_key_usage_oids`	array	no	A comma-separated string or list of extended key usage oids.
`generate_lease`	boolean	no	If set, certificates issued/signed against this role will have Vault leases attached to them. Defaults to “false”. Certificates can be

Parameter	Type	Required	Description
			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.
`issuer_ref`	string	no	Reference to the issuer used to sign requests serviced by this role.
`key_bits`	integer	no	The number of bits to use. Allowed values are 0 (universal default); with rsa key_type: 2048 (default), 3072, or 4096; with ec key_type: 224, 256 (default), 384, or 521; ignored with ed25519.
`key_type`	string	no	The type of key to use; defaults to RSA. “rsa” “ec”, “ed25519”, “gost3410-256-paramset-a”, “gost3410-256-paramset-b”, “gost3410-256-paramset-c”, “gost3410-256-paramset-d”, “gost3410-512-paramset-a”, “gost3410-512-paramset-b”, “gost3410-512-paramset-c” and “any” are the only valid values.
`key_usage`	array	no	A comma-separated string or list of key usages (not extended key usages). Valid values can be found at https://golang.org/pkg/

Parameter	Type	Required	Description
			crypto/x509/#KeyUsage – simply drop the “KeyUsage” part of the name. To remove all key usages from being set, set this value to an empty list. See also RFC 5280 Section 4.2.1.3.
`locality`	array	no	If set, Locality will be set to this value in certificates issued by this role.
`max_ttl`	integer	no	The maximum allowed lease duration. If not set, defaults to the system maximum lease TTL.
`no_store`	boolean	no	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”.
`not_after`	string	no	Set the not after field of the certificate with specified date value. The value format should be given in UTC format YYYY-MM-ddTHH:MM:SSZ.
`not_before_duration`	integer	no	The duration in seconds before now which the certificate needs to be backdated by.
`organization`	array	no	If set, O (Organization) will be set to this value in certificates issued by this role.

Parameter	Type	Required	Description
`ou`	array	no	If set, OU (OrganizationalUnit) will be set to this value in certificates issued by this role.
`policy_identifiers`	array	no	A comma-separated string or list of policy OIDs, or a JSON list of qualified policy information, which must include an oid, and may include a notice and/or cps url, using the form [{“oid”=“1.3.6.1.4.1.7.8”,“notice”=“I am a user Notice”}, {“oid”=“1.3.6.1.4.1.44947.1.2.4 “,“cps”=“https://example.com”}].
`postal_code`	array	no	If set, Postal Code will be set to this value in certificates issued by this role.
`province`	array	no	If set, Province will be set to this value in certificates issued by this role.
`require_cn`	boolean	no	If set to false, makes the ‘common_name’ field optional while generating a certificate.
`server_flag`	boolean (default: True)	no	If set, certificates are flagged for server auth use. Defaults to true. See also RFC 5280 Section 4.2.1.12.
`signature_bits`	integer	no	The number of bits to use in the signature algorithm; accepts 256 for SHA-2-256, 384 for SHA-2-384, and 512 for SHA-2-512. Defaults to 0 to automatically detect based on key length (SHA-2-256 for RSA keys, and matching the curve size for NIST P-Curves).
`single_valued_rdns`		no	

Parameter	Type	Required	Description
	boolean (default: False)		If set to true, each subject attribute value (Country, Organization, OU, etc.) is encoded as its own separate single-valued RDN in the certificate. This matches the common practice of most CAs (OpenSSL, Microsoft CA). When false, all values of the same attribute type are grouped into a single multi-valued RDN.
`street_address`	array	no	If set, Street Address will be set to this value in certificates issued by this role.
`ttl`	integer	no	The lease duration (validity period of the certificate) if no specific lease duration is requested. The lease duration controls the expiration of certificates issued by this backend. Defaults to the system default value or the value of max_ttl, whichever is shorter.
`use_csr_common_name`	boolean	no	If set, when used with a signing profile, the common name in the CSR will be used. This does *not* include any requested Subject Alternative Names; use use_csr_sans for that. Defaults to true.
`use_csr_sans`	boolean	no	If set, when used with a signing profile, the SANs in the CSR will be used. This does *not* include the Common Name (cn); use use_csr_common_name for that. Defaults to true.
`use_pss`	boolean	no	

Parameter	Type	Required	Description
			Whether or not to use PSS signatures when using a RSA key-type issuer. Defaults to false.

DELETE /{pki_mount_path}/roles/{name}

Operation ID: `pki-delete-role`

Manage the roles that can be created with this backend.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the role
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: No Content

POST /{pki_mount_path}/roles/{role}/acme/account/{kid}

Operation ID: `pki-write-roles-role-acme-account-kid`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`kid`	string	path	yes	The key identifier provided by the CA
`role`	string	path	yes	The desired role for the acme request
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request ‘payload’ value
`protected`	string	no	ACME request ‘protected’ value
`signature`	string	no	ACME request ‘signature’ value

Responses

200: OK

POST /{pki_mount_path}/roles/{role}/acme/authorization/{auth_id}

Operation ID: `pki-write-roles-role-acme-authorization-auth_id`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`auth_id`	string	path	yes	ACME authorization identifier value
`role`	string	path	yes	The desired role for the acme request
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request ‘payload’ value
`protected`	string	no	ACME request ‘protected’ value
`signature`	string	no	ACME request ‘signature’ value

Responses

200: OK

POST /{pki_mount_path}/roles/{role}/acme/challenge/{auth_id}/{challenge_type}

Operation ID: `pki-write-roles-role-acme-challenge-auth_id-challenge_type`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`auth_id`	string	path	yes	ACME authorization identifier value
`challenge_type`	string	path	yes	ACME challenge type
`role`	string	path	yes	The desired role for the acme request
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request ‘payload’ value
`protected`	string	no	ACME request ‘protected’ value
`signature`	string	no	ACME request ‘signature’ value

Responses

200: OK

GET /{pki_mount_path}/roles/{role}/acme/directory

Operation ID: `pki-read-roles-role-acme-directory`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`role`	string	path	yes	The desired role for the acme request
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /{pki_mount_path}/roles/{role}/acme/new-account

Operation ID: `pki-write-roles-role-acme-new-account`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`role`	string	path	yes	The desired role for the acme request
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request ‘payload’ value
`protected`	string	no	ACME request ‘protected’ value
`signature`	string	no	ACME request ‘signature’ value

Responses

200: OK

POST /{pki_mount_path}/roles/{role}/acme/new-eab

Operation ID: `pki-generate-eab-key-for-role`

Generate external account bindings to be used for ACME

Parameters

Parameter	Type	Location	Required	Description
`role`	string	path	yes	The desired role for the acme request
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`acme_directory`	string	no	The ACME directory to which the key belongs
`created_on`	string	no	An RFC3339 formatted date time when the EAB token was created
`id`	string	no	The EAB key identifier
`key`	string	no	The EAB hmac key
`key_type`	string	no	The EAB key type

GET /{pki_mount_path}/roles/{role}/acme/new-nonce

Operation ID: `pki-read-roles-role-acme-new-nonce`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`role`	string	path	yes	The desired role for the acme request
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /{pki_mount_path}/roles/{role}/acme/new-order

Operation ID: `pki-write-roles-role-acme-new-order`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`role`	string	path	yes	The desired role for the acme request
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request ‘payload’ value
`protected`	string	no	ACME request ‘protected’ value
`signature`	string	no	ACME request ‘signature’ value

Responses

200: OK

POST /{pki_mount_path}/roles/{role}/acme/order/{order_id}

Operation ID: `pki-write-roles-role-acme-order-order_id`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`order_id`	string	path	yes	The ACME order identifier to fetch
`role`	string	path	yes	The desired role for the acme request
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request ‘payload’ value
`protected`	string	no	ACME request ‘protected’ value
`signature`	string	no	ACME request ‘signature’ value

Responses

200: OK

POST /{pki_mount_path}/roles/{role}/acme/order/{order_id}/cert

Operation ID: `pki-write-roles-role-acme-order-order_id-cert`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`order_id`	string	path	yes	The ACME order identifier to fetch
`role`	string	path	yes	The desired role for the acme request
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request ‘payload’ value
`protected`	string	no	ACME request ‘protected’ value
`signature`	string	no	ACME request ‘signature’ value

Responses

200: OK

POST /{pki_mount_path}/roles/{role}/acme/order/{order_id}/finalize

Operation ID: `pki-write-roles-role-acme-order-order_id-finalize`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`order_id`	string	path	yes	The ACME order identifier to fetch
`role`	string	path	yes	The desired role for the acme request
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request ‘payload’ value
`protected`	string	no	ACME request ‘protected’ value
`signature`	string	no	ACME request ‘signature’ value

Responses

200: OK

POST /{pki_mount_path}/roles/{role}/acme/orders

Operation ID: `pki-write-roles-role-acme-orders`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`role`	string	path	yes	The desired role for the acme request
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request ‘payload’ value
`protected`	string	no	ACME request ‘protected’ value
`signature`	string	no	ACME request ‘signature’ value

Responses

200: OK

POST /{pki_mount_path}/roles/{role}/acme/revoke-cert

Operation ID: `pki-write-roles-role-acme-revoke-cert`

An endpoint implementing the standard ACME protocol

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`role`	string	path	yes	The desired role for the acme request
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`payload`	string	no	ACME request ‘payload’ value
`protected`	string	no	ACME request ‘protected’ value
`signature`	string	no	ACME request ‘signature’ value

Responses

200: OK

DELETE /{pki_mount_path}/root

Operation ID: `pki-delete-root`

Deletes the root CA key to allow a new one to be generated.

Required sudo: yes

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /{pki_mount_path}/root/generate/{exported}

Operation ID: `pki-generate-root`

Generate a new CA certificate and private key used for signing.

Parameters

| Parameter | Type | Location | Required | Description |

| ———- | —- | ——— | ——— | ———- | | `exported` | string (internal, exported, kms) | path | yes | Must be “internal”, “exported” or “kms”. If set to “exported”, the generated private key will be

returned. This is your *only* chance to retrieve the private key! | | `pki_mount_path` | string | path | yes | Path that the backend was mounted at |

Request body parameters

Parameter	Type	Required	Description
`alt_names`	string	no	The requested Subject Alternative Names, if any, in a comma-delimited list. May contain both DNS names and email addresses.
`common_name`	string	no	The requested common name; if you want more than one, specify the alternative names in the alt_names map. If not specified when signing, the common name will be taken from the CSR; other names must still be specified in alt_names or ip_sans.
`country`	array	no	If set, Country will be set to this value.
`exclude_cn_from_sans`	boolean (default: False)	no	If true, the Common Name will not be included in DNS or Email Subject Alternate Names. Defaults to false (CN is included).
`format`	string (pem, der, pem_bundle) (default: pem)	no	Format for returned data. Can be “pem”, “der”, or “pem_bundle”. If “pem_bundle”, any private key and issuing cert will be appended to the certificate pem. If “der”, the value will be base64 encoded. Defaults to “pem”.
`ip_sans`	array	no	The requested IP SANs, if any, in a comma-delimited list
`issuer_name`	string	no	Provide a name to the generated or existing issuer, the name must be unique across all issuers and not be the reserved value ‘default’
`key_bits`	integer (default: 0)	no	The number of bits to use. Allowed values are 0 (universal default); with rsa key_type: 2048 (default), 3072, or 4096; with ec key_type:

Parameter	Type	Required	Description
			224, 256 (default), 384, or 521; ignored with ed25519.
`key_name`	string	no	Provide a name to the generated or existing key, the name must be unique across all keys and not be the reserved value ‘default’
`key_ref`	string (default: default)	no	Reference to a existing key; either “default” for the configured default key, an identifier or the name assigned to the key.
`key_type`	string (rsa, ec, ed25519, gost3410-256-paramset-a, gost3410-256-paramset-b, gost3410-256-paramset-c, gost3410-256-paramset-d, gost3410-512-paramset-a, gost3410-512-paramset-b, gost3410-512-paramset-c) (default: rsa)	no	The type of key to use; defaults to RSA. “rsa” “ec”, “ed25519”, “gost3410-256-paramset-a”, “gost3410-256-paramset-b”, “gost3410-256-paramset-c”, “gost3410-256-paramset-d”, “gost3410-512-paramset-a”, “gost3410-512-paramset-b”, “gost3410-512-paramset-c” are the only valid values.
`locality`	array	no	If set, Locality will be set to this value.
`managed_key_id`	string	no	The name of the managed key to use when the exported type is kms. When kms type is the key type, this field or managed_key_name is required. Ignored for other types.
`managed_key_name`	string	no	The name of the managed key to use when the exported type is

Parameter	Type	Required	Description
			kms. When kms type is the key type, this field or managed_key_id is required. Ignored for other types.
`max_path_length`	integer (default: -1)	no	The maximum allowable path length
`not_after`	string	no	Set the not after field of the certificate with specified date value. The value format should be given in UTC format YYYY-MM-ddTHH:MM:SSZ
`not_before_duration`	integer (default: 30)	no	The duration before now which the certificate needs to be backdated by.
`organization`	array	no	If set, O (Organization) will be set to this value.
`other_sans`	array	no	Requested other SANs, in an array with the format ;UTF8: for each entry.
`ou`	array	no	If set, OU (OrganizationalUnit) will be set to this value.
`permitted_dns_domains`	array	no	Domains for which this certificate is allowed to sign or issue child certificates. If set, all DNS names (subject and alt) on child certs must be exact matches or subsets of the given domains (see https://tools.ietf.org/html/rfc5280#section-4.2.1.10).
`postal_code`	array	no	If set, Postal Code will be set to this value.
`private_key_format`	string (, der, pem, pkcs8) (default: der)	no	Format for the returned private key. Generally the default will be controlled by the “format” parameter as either base64-encoded DER or PEM-encoded

Parameter	Type	Required	Description
			DER. However, this can be set to “pkcs8” to have the returned private key contain base64-encoded pkcs8 or PEM-encoded pkcs8 instead. Defaults to “der”.
`province`	array	no	If set, Province will be set to this value.
`serial_number`	string	no	The Subject’s requested serial number, if any. See RFC 4519 Section 2.31 ‘serialNumber’ for a description of this field. If you want more than one, specify alternative names in the alt_names map using OID 2.5.4.5. This has no impact on the final certificate’s Serial Number field.
`signature_bits`	integer (default: 0)	no	The number of bits to use in the signature algorithm; accepts 256 for SHA-2-256, 384 for SHA-2-384, and 512 for SHA-2-512. Defaults to 0 to automatically detect based on key length (SHA-2-256 for RSA keys, and matching the curve size for NIST P-Curves).
`street_address`	array	no	If set, Street Address will be set to this value.
`ttl`	integer	no	The requested Time To Live for the certificate; sets the expiration date. If not specified the role default, backend default, or system default TTL is used, in that order. Cannot be larger than the mount max TTL. Note: this only has an effect when generating a CA cert or signing a CA cert, not when generating a CSR for an intermediate CA.

Parameter	Type	Required	Description
`uri_sans`	array	no	The requested URI SANs, if any, in a comma-delimited list.
`use_pss`	boolean (default: False)	no	Whether or not to use PSS signatures when using a RSA key-type issuer. Defaults to false.

Responses

200: OK

Parameter	Type	Required	Description
`certificate`	string	no	The generated self-signed CA certificate.
`expiration`	integer	no	The expiration of the given issuer.
`issuer_id`	string	no	The ID of the issuer
`issuer_name`	string	no	The name of the issuer.
`issuing_ca`	string	no	The issuing certificate authority.
`key_id`	string	no	The ID of the key.
`key_name`	string	no	The key name if given.
`private_key`	string	no	The private key if exported was specified.
`serial_number`	string	no	The requested Subject's named serial number.

POST /{pki_mount_path}/root/replace

Operation ID: `pki-replace-root`

Read and set the default issuer certificate for signing.

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`default`	string (default: next)	no	Reference (name or identifier) to the default issuer.

Responses

200: OK

Parameter	Type	Required	Description
`default`	string	no	Reference (name or identifier) to the default issuer.
`default_follows_latest_issuer`	boolean	no	Whether the default issuer should automatically follow the latest generated or imported issuer. Defaults to false.

POST /{pki_mount_path}/root/rotate/{exported}

Operation ID: `pki-rotate-root`

Generate a new CA certificate and private key used for signing.

Parameters

| Parameter | Type | Location | Required | Description |

| ———- | —- | ——— | ——— | ———- | | `exported` | string (internal, exported, kms) | path | yes | Must be “internal”, “exported” or “kms”. If set to “exported”, the generated private key will be returned. This is your *only* chance to retrieve the private key! | | `pki_mount_path` | string | path | yes | Path that the backend was mounted at |

Request body parameters

Parameter	Type	Required	Description
`alt_names`	string	no	The requested Subject Alternative Names, if any, in a comma-delimited list. May contain both DNS names and email addresses.
`common_name`	string	no	The requested common name; if you want more than one, specify the alternative names in the alt_names map. If not specified when signing, the common name will be taken from the CSR; other names must still be specified in alt_names or ip_sans.
`country`	array	no	If set, Country will be set to this value.
`exclude_cn_from_sans`	boolean (default: False)	no	If true, the Common Name will not be included in DNS or Email Subject Alternate Names. Defaults to false (CN is included).
`format`	string (pem, der, pem_bundle) (default: pem)	no	Format for returned data. Can be “pem”, “der”, or “pem_bundle”. If “pem_bundle”, any private key and issuing cert will be appended to the certificate pem. If “der”, the value will be base64 encoded. Defaults to “pem”.
`ip_sans`	array	no	The requested IP SANs, if any, in a comma-delimited list
`issuer_name`	string	no	Provide a name to the generated or existing issuer, the name must be unique across all issuers and not be the reserved value ‘default’
`key_bits`	integer (default: 0)	no	The number of bits to use. Allowed values are 0 (universal default); with rsa key_type: 2048 (default), 3072, or 4096; with ec key_type:

Parameter	Type	Required	Description
			224, 256 (default), 384, or 521; ignored with ed25519.
`key_name`	string	no	Provide a name to the generated or existing key, the name must be unique across all keys and not be the reserved value ‘default’
`key_ref`	string (default: default)	no	Reference to a existing key; either “default” for the configured default key, an identifier or the name assigned to the key.
`key_type`	string (rsa, ec, ed25519, gost3410-256-paramset-a, gost3410-256-paramset-b, gost3410-256-paramset-c, gost3410-256-paramset-d, gost3410-512-paramset-a, gost3410-512-paramset-b, gost3410-512-paramset-c) (default: rsa)	no	The type of key to use; defaults to RSA. “rsa” “ec”, “ed25519”, “gost3410-256-paramset-a”, “gost3410-256-paramset-b”, “gost3410-256-paramset-c”, “gost3410-256-paramset-d”, “gost3410-512-paramset-a”, “gost3410-512-paramset-b”, “gost3410-512-paramset-c” are the only valid values.
`locality`	array	no	If set, Locality will be set to this value.
`managed_key_id`	string	no	The name of the managed key to use when the exported type is kms. When kms type is the key type, this field or managed_key_name is required. Ignored for other types.
`managed_key_name`	string	no	The name of the managed key to use when the exported type is

Parameter	Type	Required	Description
			kms. When kms type is the key type, this field or managed_key_id is required. Ignored for other types.
`max_path_length`	integer (default: -1)	no	The maximum allowable path length
`not_after`	string	no	Set the not after field of the certificate with specified date value. The value format should be given in UTC format YYYY-MM-ddTHH:MM:SSZ
`not_before_duration`	integer (default: 30)	no	The duration before now which the certificate needs to be backdated by.
`organization`	array	no	If set, O (Organization) will be set to this value.
`other_sans`	array	no	Requested other SANs, in an array with the format ;UTF8: for each entry.
`ou`	array	no	If set, OU (OrganizationalUnit) will be set to this value.
`permitted_dns_domains`	array	no	Domains for which this certificate is allowed to sign or issue child certificates. If set, all DNS names (subject and alt) on child certs must be exact matches or subsets of the given domains (see https://tools.ietf.org/html/rfc5280#section-4.2.1.10).
`postal_code`	array	no	If set, Postal Code will be set to this value.
`private_key_format`	string (, der, pem, pkcs8) (default: der)	no	Format for the returned private key. Generally the default will be controlled by the “format” parameter as either base64-encoded DER or PEM-encoded

Parameter	Type	Required	Description
			DER. However, this can be set to “pkcs8” to have the returned private key contain base64-encoded pkcs8 or PEM-encoded pkcs8 instead. Defaults to “der”.
`province`	array	no	If set, Province will be set to this value.
`serial_number`	string	no	The Subject’s requested serial number, if any. See RFC 4519 Section 2.31 ‘serialNumber’ for a description of this field. If you want more than one, specify alternative names in the alt_names map using OID 2.5.4.5. This has no impact on the final certificate’s Serial Number field.
`signature_bits`	integer (default: 0)	no	The number of bits to use in the signature algorithm; accepts 256 for SHA-2-256, 384 for SHA-2-384, and 512 for SHA-2-512. Defaults to 0 to automatically detect based on key length (SHA-2-256 for RSA keys, and matching the curve size for NIST P-Curves).
`street_address`	array	no	If set, Street Address will be set to this value.
`ttl`	integer	no	The requested Time To Live for the certificate; sets the expiration date. If not specified the role default, backend default, or system default TTL is used, in that order. Cannot be larger than the mount max TTL. Note: this only has an effect when generating a CA cert or signing a CA cert, not when generating a CSR for an intermediate CA.

Parameter	Type	Required	Description
`uri_sans`	array	no	The requested URI SANs, if any, in a comma-delimited list.
`use_pss`	boolean (default: False)	no	Whether or not to use PSS signatures when using a RSA key-type issuer. Defaults to false.

Responses

200: OK

Parameter	Type	Required	Description
`certificate`	string	no	The generated self-signed CA certificate.
`expiration`	integer	no	The expiration of the given issuer.
`issuer_id`	string	no	The ID of the issuer
`issuer_name`	string	no	The name of the issuer.
`issuing_ca`	string	no	The issuing certificate authority.
`key_id`	string	no	The ID of the key.
`key_name`	string	no	The key name if given.
`private_key`	string	no	The private key if exported was specified.
`serial_number`	string	no	The requested Subject's named serial number.

POST /{pki_mount_path}/root/sign-intermediate

Operation ID: `pki-root-sign-intermediate`

Issue an intermediate CA certificate based on the provided CSR.

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`alt_names`	string	no	The requested Subject Alternative Names, if any, in a comma-delimited list. May contain both DNS names and email addresses.
`common_name`	string	no	The requested common name; if you want more than one, specify the alternative names in the alt_names map. If not specified when signing, the common name will be taken from the CSR; other names must still be specified in alt_names or ip_sans.
`country`	array	no	If set, Country will be set to this value.
`csr`	string (default:)	no	PEM-format CSR to be signed.
`exclude_cn_from_sans`	boolean (default: False)	no	If true, the Common Name will not be included in DNS or Email Subject Alternate Names. Defaults to false (CN is included).
`format`	string (pem, der, pem_bundle) (default: pem)	no	Format for returned data. Can be “pem”, “der”, or “pem_bundle”. If “pem_bundle”, any private key and issuing cert will be appended to the certificate pem. If “der”, the value will be base64 encoded. Defaults to “pem”.
`ip_sans`	array	no	The requested IP SANs, if any, in a comma-delimited list
`issuer_name`	string	no	Provide a name to the generated or existing issuer, the name must be unique across all issuers and not be the reserved value ‘default’
`issuer_ref`	string (default: default)	no	Reference to a existing issuer; either “default” for the configured default issuer, an identifier or the name assigned to the issuer.

Parameter	Type	Required	Description
`locality`	array	no	If set, Locality will be set to this value.
`max_path_length`	integer (default: -1)	no	The maximum allowable path length
`not_after`	string	no	Set the not after field of the certificate with specified date value. The value format should be given in UTC format YYYY-MM-ddTHH:MM:SSZ
`not_before_duration`	integer (default: 30)	no	The duration before now which the certificate needs to be backdated by.
`organization`	array	no	If set, O (Organization) will be set to this value.
`other_sans`	array	no	Requested other SANs, in an array with the format ;UTF8: for each entry.
`ou`	array	no	If set, OU (OrganizationalUnit) will be set to this value.
`permitted_dns_domains`	array	no	Domains for which this certificate is allowed to sign or issue child certificates. If set, all DNS names (subject and alt) on child certs must be exact matches or subsets of the given domains (see https://tools.ietf.org/html/rfc5280#section-4.2.1.10).
`postal_code`	array	no	If set, Postal Code will be set to this value.
`private_key_format`	string (, der, pem, pkcs8) (default: der)	no	Format for the returned private key. Generally the default will be controlled by the “format” parameter as either base64-encoded DER or PEM-encoded DER. However, this can be set to “pkcs8” to have the returned private key contain base64-encoded pkcs8 or PEM-encoded pkcs8 instead. Defaults to “der”.
`province`	array	no	If set, Province will be set to this value.

Parameter	Type	Required	Description
`serial_number`	string	no	The Subject's requested serial number, if any. See RFC 4519 Section 2.31 'serialNumber' for a description of this field. If you want more than one, specify alternative names in the alt_names map using OID 2.5.4.5. This has no impact on the final certificate's Serial Number field.
`signature_bits`	integer (default: 0)	no	The number of bits to use in the signature algorithm; accepts 256 for SHA-2-256, 384 for SHA-2-384, and 512 for SHA-2-512. Defaults to 0 to automatically detect based on key length (SHA-2-256 for RSA keys, and matching the curve size for NIST P-Curves).
`skid`	string (default:)	no	Value for the Subject Key Identifier field (RFC 5280 Section 4.2.1.2). This value should ONLY be used when cross-signing to mimic the existing certificate's SKID value; this is necessary to allow certain TLS implementations (such as OpenSSL) which use SKID/AKID matches in chain building to restrict possible valid chains. Specified as a string in hex format. Default is empty, allowing Vault to automatically calculate the SKID according to method one in the above RFC section.
`street_address`	array	no	If set, Street Address will be set to this value.
`ttl`	integer	no	The requested Time To Live for the certificate; sets the expiration date. If not specified the role default, backend default, or system default TTL is used, in that order. Cannot be larger than the mount max TTL. Note: this only has an effect when generating a CA cert or

Parameter	Type	Required	Description
			signing a CA cert, not when generating a CSR for an intermediate CA.
`uri_sans`	array	no	The requested URI SANs, if any, in a comma-delimited list.
`use_csr_values`	boolean (default: False)	no	If true, then: 1) Subject information, including names and alternate names, will be preserved from the CSR rather than using values provided in the other parameters to this path; 2) Any key usages requested in the CSR will be added to the basic set of key usages used for CA certs signed by this path; for instance, the non-repudiation flag; 3) Extensions requested in the CSR will be copied into the issued certificate.
`use_pss`	boolean (default: False)	no	Whether or not to use PSS signatures when using a RSA key-type issuer. Defaults to false.

Responses

200: OK

Parameter	Type	Required	Description
`ca_chain`	array	no	CA Chain
`certificate`	string	no	Certificate
`expiration`	integer	no	Expiration Time
`issuing_ca`	string	no	Issuing CA
`serial_number`	string	no	Serial Number

POST /{pki_mount_path}/root/sign-self-issued

Operation ID: `pki-root-sign-self-issued`

Re-issue a self-signed certificate based on the provided certificate.

Required sudo: yes

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`certificate`	string	no	PEM-format self-issued certificate to be signed.
`issuer_ref`	string (default: default)	no	Reference to a existing issuer; either “default” for the configured default issuer, an identifier or the name assigned to the issuer.
`require_matching_certificate_algorithms`	boolean (default: False)	no	If true, require the public key algorithm of the signer to match that of the self issued certificate.

Responses

200: OK

Parameter	Type	Required	Description
`certificate`	string	no	Certificate
`issuing_ca`	string	no	Issuing CA

POST /{pki_mount_path}/sign-verbatim

Operation ID: `pki-sign-verbatim`

Issue a certificate directly based on the provided CSR.

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`alt_names`	string	no	The requested Subject Alternative Names, if any, in a comma-delimited list. If email protection is enabled for the role, this may contain email addresses.
`common_name`	string	no	The requested common name; if you want more than one, specify the alternative names in the alt_names map. If email protection is enabled in the role, this may be an email address.
`csr`	string (default:)	no	PEM-format CSR to be signed. Values will be taken verbatim from the CSR, except for basic constraints.
`exclude_cn_from_sans`	boolean (default: False)	no	If true, the Common Name will not be included in DNS or Email Subject Alternate Names. Defaults to false (CN is included).
`ext_key_usage`	array (default: [])	no	A comma-separated string or list of extended key usages. Valid values can be found at https://golang.org/pkg/crypto/x509/#ExtKeyUsage – simply drop the “ExtKeyUsage” part of the name. To remove all key usages from being set, set this value to an empty list.
`ext_key_usage_oids`	array	no	A comma-separated string or list of extended key usage oids.
`format`	string (pem, der, pem_bundle) (default: pem)	no	Format for returned data. Can be “pem”, “der”, or “pem_bundle”. If “pem_bundle”, any private key and issuing cert

Parameter	Type	Required	Description
			will be appended to the certificate pem. If “der”, the value will be base64 encoded. Defaults to “pem”.
`ip_sans`	array	no	The requested IP SANs, if any, in a comma-delimited list
`issuer_ref`	string (default: default)	no	Reference to a existing issuer; either “default” for the configured default issuer, an identifier or the name assigned to the issuer.
`key_usage`	array (default: [‘DigitalSignature’, ‘KeyAgreement’, ‘KeyEncipherment’])	no	A comma-separated string or list of key usages (not extended key usages). Valid values can be found at https://golang.org/pkg/crypto/x509/#KeyUsage – simply drop the “KeyUsage” part of the name. To remove all key usages from being set, set this value to an empty list.
`not_after`	string	no	Set the not after field of the certificate with specified date value. The value format should be given in UTC format YYYY-MM-ddTHH:MM:SSZ
`other_sans`	array	no	Requested other SANs, in an array with the format ;UTF8: for each entry.
`private_key_format`	string (, der, pem, pkcs8) (default: der)	no	Format for the returned private key. Generally the default will be controlled by the “format” parameter as either base64-encoded DER or PEM-encoded DER. However, this can be set to “pkcs8” to have the returned private key contain base64-encoded pkcs8 or PEM-

Parameter	Type	Required	Description
			encoded pkcs8 instead. Defaults to “der”.
`remove_roots_from_chain`	boolean (default: False)	no	Whether or not to remove self-signed CA certificates in the output of the ca_chain field.
`role`	string	no	The desired role with configuration for this request
`serial_number`	string	no	The Subject’s requested serial number, if any. See RFC 4519 Section 2.31 ‘serialNumber’ for a description of this field. If you want more than one, specify alternative names in the alt_names map using OID 2.5.4.5. This has no impact on the final certificate’s Serial Number field.
`signature_bits`	integer (default: 0)	no	The number of bits to use in the signature algorithm; accepts 256 for SHA-2-256, 384 for SHA-2-384, and 512 for SHA-2-512. Defaults to 0 to automatically detect based on key length (SHA-2-256 for RSA keys, and matching the curve size for NIST P-Curves).
`ttl`	integer	no	The requested Time To Live for the certificate; sets the expiration date. If not specified the role default, backend default, or system default TTL is used, in that order. Cannot be larger than the role max TTL.
`uri_sans`	array	no	The requested URI SANs, if any, in a comma-delimited list.
`use_pss`		no	

Parameter	Type	Required	Description
	boolean (default: False)		Whether or not to use PSS signatures when using a RSA key-type issuer. Defaults to false.
`user_ids`	array	no	The requested user_ids value to place in the subject, if any, in a comma-delimited list. Restricted by allowed_user_ids. Any values are added with OID 0.9.2342.19200300.100.1.1.

Responses

200: OK

Parameter	Type	Required	Description
`ca_chain`	array	no	Certificate Chain
`certificate`	string	no	Certificate
`expiration`	integer	no	Time of expiration
`issuing_ca`	string	no	Issuing Certificate Authority
`private_key`	string	no	Private key
`private_key_type`	string	no	Private key type
`serial_number`	string	no	Serial Number

POST /{pki_mount_path}/sign-verbatim/{role}

Operation ID: `pki-sign-verbatim-with-role`

Issue a certificate directly based on the provided CSR.

Parameters

Parameter	Type	Location	Required	Description
`role`	string	path	yes	The desired role with configuration for this request
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`alt_names`	string	no	The requested Subject Alternative Names, if any, in a comma-delimited list. If email protection is enabled for the role, this may contain email addresses.
`common_name`	string	no	The requested common name; if you want more than one, specify the alternative names in the alt_names map. If email protection is enabled in the role, this may be an email address.
`csr`	string (default:)	no	PEM-format CSR to be signed. Values will be taken verbatim from the CSR, except for basic constraints.
`exclude_cn_from_sans`	boolean (default: False)	no	If true, the Common Name will not be included in DNS or Email Subject Alternate Names. Defaults to false (CN is included).
`ext_key_usage`	array (default: [])	no	A comma-separated string or list of extended key usages. Valid values can be found at https://golang.org/pkg/crypto/x509/#ExtKeyUsage – simply drop the “ExtKeyUsage” part of the name. To remove all key usages from being set, set this value to an empty list.
`ext_key_usage_oids`	array	no	A comma-separated string or list of extended key usage oids.
`format`	string (pem, der, pem_bundle) (default: pem)	no	Format for returned data. Can be “pem”, “der”, or “pem_bundle”. If “pem_bundle”, any private key and issuing cert

Parameter	Type	Required	Description
			will be appended to the certificate pem. If “der”, the value will be base64 encoded. Defaults to “pem”.
`ip_sans`	array	no	The requested IP SANs, if any, in a comma-delimited list
`issuer_ref`	string (default: default)	no	Reference to a existing issuer; either “default” for the configured default issuer, an identifier or the name assigned to the issuer.
`key_usage`	array (default: [‘DigitalSignature’, ‘KeyAgreement’, ‘KeyEncipherment’])	no	A comma-separated string or list of key usages (not extended key usages). Valid values can be found at https://golang.org/pkg/crypto/x509/#KeyUsage – simply drop the “KeyUsage” part of the name. To remove all key usages from being set, set this value to an empty list.
`not_after`	string	no	Set the not after field of the certificate with specified date value. The value format should be given in UTC format YYYY-MM-ddTHH:MM:SSZ
`other_sans`	array	no	Requested other SANs, in an array with the format ;UTF8: for each entry.
`private_key_format`	string (, der, pem, pkcs8) (default: der)	no	Format for the returned private key. Generally the default will be controlled by the “format” parameter as either base64-encoded DER or PEM-encoded DER. However, this can be set to “pkcs8” to have the returned private key contain base64-encoded pkcs8 or PEM-

Parameter	Type	Required	Description
			encoded pkcs8 instead. Defaults to “der”.
`remove_roots_from_chain`	boolean (default: False)	no	Whether or not to remove self-signed CA certificates in the output of the ca_chain field.
`serial_number`	string	no	The Subject’s requested serial number, if any. See RFC 4519 Section 2.31 ‘serialNumber’ for a description of this field. If you want more than one, specify alternative names in the alt_names map using OID 2.5.4.5. This has no impact on the final certificate’s Serial Number field.
`signature_bits`	integer (default: 0)	no	The number of bits to use in the signature algorithm; accepts 256 for SHA-2-256, 384 for SHA-2-384, and 512 for SHA-2-512. Defaults to 0 to automatically detect based on key length (SHA-2-256 for RSA keys, and matching the curve size for NIST P-Curves).
`ttl`	integer	no	The requested Time To Live for the certificate; sets the expiration date. If not specified the role default, backend default, or system default TTL is used, in that order. Cannot be larger than the role max TTL.
`uri_sans`	array	no	The requested URI SANs, if any, in a comma-delimited list.
`use_pss`	boolean (default: False)	no	Whether or not to use PSS signatures when using a RSA

Parameter	Type	Required	Description
			key-type issuer. Defaults to false.
`user_ids`	array	no	The requested user_ids value to place in the subject, if any, in a comma-delimited list. Restricted by allowed_user_ids. Any values are added with OID 0.9.2342.19200300.100.1.1.

Responses

200: OK

Parameter	Type	Required	Description
`ca_chain`	array	no	Certificate Chain
`certificate`	string	no	Certificate
`expiration`	integer	no	Time of expiration
`issuing_ca`	string	no	Issuing Certificate Authority
`private_key`	string	no	Private key
`private_key_type`	string	no	Private key type
`serial_number`	string	no	Serial Number

POST /{pki_mount_path}/sign/{role}

Operation ID: `pki-sign-with-role`

Request certificates using a certain role with the provided details.

Parameters

Parameter	Type	Location	Required	Description
`role`	string	path	yes	The desired role with configuration for this request
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`alt_names`	string	no	The requested Subject Alternative Names, if any, in a comma-delimited list. If email protection is enabled for the role, this may contain email addresses.
`common_name`	string	no	The requested common name; if you want more than one, specify the alternative names in the alt_names map. If email protection is enabled in the role, this may be an email address.
`csr`	string (default:)	no	PEM-format CSR to be signed.
`exclude_cn_from_sans`	boolean (default: False)	no	If true, the Common Name will not be included in DNS or Email Subject Alternate Names. Defaults to false (CN is included).
`format`	string (pem, der, pem_bundle) (default: pem)	no	Format for returned data. Can be “pem”, “der”, or “pem_bundle”. If “pem_bundle”, any private key and issuing cert will be appended to the certificate pem. If “der”, the value will be base64 encoded. Defaults to “pem”.
`ip_sans`	array	no	The requested IP SANs, if any, in a comma-delimited list
`issuer_ref`	string (default: default)	no	Reference to a existing issuer; either “default” for the configured default issuer, an identifier or the name assigned to the issuer.
`not_after`	string	no	Set the not after field of the certificate with specified date value. The value format should be given in UTC format YYYY-MM-ddTHH:MM:SSZ
`other_sans`	array	no	Requested other SANs, in an array with the format ;UTF8: for each entry.

Parameter	Type	Required	Description
`private_key_format`	string (, der, pem, pkcs8) (default: der)	no	Format for the returned private key. Generally the default will be controlled by the “format” parameter as either base64-encoded DER or PEM-encoded DER. However, this can be set to “pkcs8” to have the returned private key contain base64-encoded pkcs8 or PEM-encoded pkcs8 instead. Defaults to “der”.
`remove_roots_from_chain`	boolean (default: False)	no	Whether or not to remove self-signed CA certificates in the output of the ca_chain field.
`serial_number`	string	no	The Subject’s requested serial number, if any. See RFC 4519 Section 2.31 ‘serialNumber’ for a description of this field. If you want more than one, specify alternative names in the alt_names map using OID 2.5.4.5. This has no impact on the final certificate’s Serial Number field.
`ttl`	integer	no	The requested Time To Live for the certificate; sets the expiration date. If not specified the role default, backend default, or system default TTL is used, in that order. Cannot be larger than the role max TTL.
`uri_sans`	array	no	The requested URI SANs, if any, in a comma-delimited list.
`user_ids`	array	no	The requested user_ids value to place in the subject, if any, in a comma-delimited list. Restricted by allowed_user_ids. Any values are added with OID 0.9.2342.19200300.100.1.1.

Responses

200: OK

Parameter	Type	Required	Description
`ca_chain`	array	no	Certificate Chain
`certificate`	string	no	Certificate
`expiration`	integer	no	Time of expiration
`issuing_ca`	string	no	Issuing Certificate Authority
`private_key`	string	no	Private key
`private_key_type`	string	no	Private key type
`serial_number`	string	no	Serial Number

POST /{pki_mount_path}/tidy

Operation ID: `pki-tidy`

Tidy up the backend by removing expired certificates, revocation information, or both.

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`acme_account_safety_buffer`	integer (default: 2592000)	no	The amount of time that must pass after creation that an account with no orders is marked revoked, and the amount of time after being marked revoked or deactivated.
`issuer_safety_buffer`	integer (default: 31536000)	no	The amount of extra time that must have passed beyond issuer's expiration before it is removed from the backend storage. Defaults to 8760 hours (1 year).
`pause_duration`	string (default: 0s)	no	The amount of time to wait between processing certificates. This allows operators to change the execution profile of tidy to take consume less resources by slowing down how long it takes to run. Note that the entire list of certificates will be stored in memory during the entire tidy operation, but resources to read/process/update existing entries will be spread out over a greater period of time. By default this is zero seconds.
`revocation_queue_safety_buffer`	integer (default: 172800)	no	The amount of time that must pass from the cross-cluster revocation request being initiated to when it will be slated for removal. Setting this too low may remove valid revocation requests before the owning

Parameter	Type	Required	Description
			cluster has a chance to process them, especially if the cluster is offline.
`safety_buffer`	integer (default: 259200)	no	The amount of extra time that must have passed beyond certificate expiration before it is removed from the backend storage and/or revocation list. Defaults to 72 hours.
`tidy_acme`	boolean (default: False)	no	Set to true to enable tidying ACME accounts, orders and authorizations. ACME orders are tidied (deleted) safety_buffer after the certificate associated with them expires, or after the order and relevant authorizations have expired if no certificate was produced. Authorizations are tidied with the corresponding order. When a valid ACME Account is at least acme_account_safety_buffer old, and has no remaining orders associated with it, the account is marked as revoked. After another acme_account_safety_buffer has passed from the revocation or deactivation date, a revoked or deactivated ACME account is deleted.
`tidy_cert_store`	boolean	no	Set to true to enable tidying up the certificate store
`tidy_cross_cluster_revoked_certs`	boolean	no	

Parameter	Type	Required	Description
			Set to true to enable tidying up the cross-cluster revoked certificate store. Only runs on the active primary node.
`tidy_expired_issuers`	boolean	no	Set to true to automatically remove expired issuers past the issuer_safety_buffer. No keys will be removed as part of this operation.
`tidy_move_legacy_ca_bundle`	boolean	no	Set to true to move the legacy ca_bundle from /config/ca_bundle to /config/ca_bundle.bak. This prevents downgrades to pre-Vault 1.11 versions (as older PKI engines do not know about the new multi-issuer storage layout), but improves the performance on seal wrapped PKI mounts. This will only occur if at least issuer_safety_buffer time has occurred after the initial storage migration. This backup is saved in case of an issue in future migrations. Operators may consider removing it via sys/raw if they desire. The backup will be removed via a DELETE /root call, but note that this removes ALL issuers within the mount (and is thus not desirable in most operational scenarios).
`tidy_revocation_list`	boolean	no	Deprecated; synonym for 'tidy_revoked_certs

Parameter	Type	Required	Description
`tidy_revocation_queue`	boolean (default: False)	no	Set to true to remove stale revocation queue entries that haven't been confirmed by any active cluster. Only runs on the active primary node
`tidy_revoked_cert_issuer_associations`	boolean	no	Set to true to validate issuer associations on revocation entries. This helps increase the performance of CRL building and OCSP responses.
`tidy_revoked_certs`	boolean	no	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.

Responses

202: Accepted

POST /{pki_mount_path}/tidy-cancel

Operation ID: `pki-tidy-cancel`

Cancels a currently running tidy operation.

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`acme_account_deleted_count`	integer	no	The number of revoked acme accounts removed
`acme_account_revoked_count`	integer	no	The number of unused acme accounts revoked
`acme_account_safety_buffer`	integer	no	Safety buffer after creation after which accounts lacking orders are revoked
`acme_orders_deleted_count`	integer	no	The number of expired, unused acme orders removed
`cert_store_deleted_count`	integer	no	The number of certificate storage entries deleted
`cross_revoked_cert_deleted_count`	integer	no	
`current_cert_store_count`	integer	no	The number of revoked certificate entries deleted
`current_revoked_cert_count`	integer	no	The number of revoked certificate entries deleted
`error`	string	no	The error message
`internal_backend_uuid`	string	no	
`issuer_safety_buffer`	integer	no	Issuer safety buffer
`last_auto_tidy_finished`	string	no	Time the last auto-tidy operation finished
`message`	string	no	Message of the operation
`missing_issuer_cert_count`	integer	no	
`pause_duration`	string	no	Duration to pause between tidying certificates
`revocation_queue_deleted_count`	integer	no	
`revocation_queue_safety_buffer`	integer	no	Revocation queue safety buffer
`revoked_cert_deleted_count`	integer	no	

Parameter	Type	Required	Description
			The number of revoked certificate entries deleted
`safety_buffer`	integer	no	Safety buffer time duration
`state`	string	no	One of Inactive, Running, Finished, or Error
`tidy_acme`	boolean	no	Tidy Unused Acme Accounts, and Orders
`tidy_cert_store`	boolean	no	Tidy certificate store
`tidy_cross_cluster_revoked_certs`	boolean	no	Tidy the cross-cluster revoked certificate store
`tidy_expired_issuers`	boolean	no	Tidy expired issuers
`tidy_move_legacy_ca_bundle`	boolean	no	
`tidy_revocation_queue`	boolean	no	
`tidy_revoked_cert_issuer_associations`	boolean	no	Tidy revoked certificate issuer associations
`tidy_revoked_certs`	boolean	no	Tidy revoked certificates
`time_finished`	string	no	Time the operation finished
`time_started`	string	no	Time the operation started
`total_acme_account_count`	integer	no	Total number of acme accounts iterated over

GET /{pki_mount_path}/tidy-status

Operation ID: `pki-tidy-status`

Returns the status of the tidy operation.

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

Parameter	Type	Required	Description
`acme_account_deleted_count`	integer	no	The number of revoked acme accounts removed
`acme_account_revoked_count`	integer	no	The number of unused acme accounts revoked
`acme_account_safety_buffer`	integer	no	Safety buffer after creation after which accounts lacking orders are revoked
`acme_orders_deleted_count`	integer	no	The number of expired, unused acme orders removed
`cert_store_deleted_count`	integer	no	The number of certificate storage entries deleted
`cross_revoked_cert_deleted_count`	integer	no	
`current_cert_store_count`	integer	no	The number of revoked certificate entries deleted
`current_revoked_cert_count`	integer	no	The number of revoked certificate entries deleted
`error`	string	no	The error message
`internal_backend_uuid`	string	no	
`issuer_safety_buffer`	integer	no	Issuer safety buffer
`last_auto_tidy_finished`	string	no	Time the last auto-tidy operation finished
`message`	string	no	Message of the operation
`missing_issuer_cert_count`	integer	no	
`pause_duration`	string	no	Duration to pause between tidying certificates
`revocation_queue_deleted_count`	integer	no	
`revocation_queue_safety_buffer`	integer	no	Revocation queue safety buffer
`revoked_cert_deleted_count`	integer	no	

Parameter	Type	Required	Description
			The number of revoked certificate entries deleted
`safety_buffer`	integer	no	Safety buffer time duration
`state`	string	no	One of Inactive, Running, Finished, or Error
`tidy_acme`	boolean	no	Tidy Unused Acme Accounts, and Orders
`tidy_cert_store`	boolean	no	Tidy certificate store
`tidy_cross_cluster_revoked_certs`	boolean	no	Tidy the cross-cluster revoked certificate store
`tidy_expired_issuers`	boolean	no	Tidy expired issuers
`tidy_move_legacy_ca_bundle`	boolean	no	
`tidy_revocation_queue`	boolean	no	
`tidy_revoked_cert_issuer_associations`	boolean	no	Tidy revoked certificate issuer associations
`tidy_revoked_certs`	boolean	no	Tidy revoked certificates
`time_finished`	string	no	Time the operation finished
`time_started`	string	no	Time the operation started
`total_acme_account_count`	integer	no	Total number of acme accounts iterated over

GET /{pki_mount_path}/unified-crl

Operation ID: `pki-read-unified-crl-der`

Fetch a CA, CRL, CA Chain, or non-revoked certificate.

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

GET /{pki_mount_path}/unified-crl/delta

Operation ID: `pki-read-unified-crl-delta`

Fetch a CA, CRL, CA Chain, or non-revoked certificate.

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

GET /{pki_mount_path}/unified-crl/delta/pem

Operation ID: `pki-read-unified-crl-delta-pem`

Fetch a CA, CRL, CA Chain, or non-revoked certificate.

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

GET /{pki_mount_path}/unified-crl/pem

Operation ID: `pki-read-unified-crl-pem`

Fetch a CA, CRL, CA Chain, or non-revoked certificate.

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /{pki_mount_path}/unified-ocsp

Operation ID: `pki-query-unified-ocsp`

Query a certificate's revocation status through OCSP'

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

GET /{pki_mount_path}/unified-ocsp/{req}

Operation ID: `pki-query-unified-ocsp-with-get-req`

Query a certificate's revocation status through OCSP'

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`req`	string	path	yes	base-64 encoded ocsp request
`pki_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /{rabbitmq_mount_path}/config/connection

Operation ID: `rabbit-mq-configure-connection`

Configure the connection URI, username, and password to talk to RabbitMQ management HTTP API.

Parameters

Parameter	Type	Location	Required	Description
`rabbitmq_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`connection_uri`	string	no	RabbitMQ Management URI
`password`	string	no	Password of the provided RabbitMQ management user
`password_policy`	string	no	Name of the password policy to use to generate passwords for dynamic credentials.
`username`	string	no	Username of a RabbitMQ management administrator
`username_template`	string	no	Template describing how dynamic usernames are generated.
`verify_connection`	boolean (default: True)	no	If set, connection_uri is verified by actually connecting to the RabbitMQ management API

Responses

200: OK

GET /{rabbitmq_mount_path}/config/lease

Operation ID: `rabbit-mq-read-lease-configuration`

Configure the lease parameters for generated credentials

Parameters

Parameter	Type	Location	Required	Description
`rabbitmq_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /{rabbitmq_mount_path}/config/lease

Operation ID: `rabbit-mq-configure-lease`

Configure the lease parameters for generated credentials

Parameters

Parameter	Type	Location	Required	Description
`rabbitmq_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`max_ttl`	integer (default: 0)	no	Duration after which the issued credentials should not be allowed to be renewed
`ttl`	integer (default: 0)	no	Duration before which the issued credentials needs renewal

Responses

200: OK

GET /{rabbitmq_mount_path}/creds/{name}

Operation ID: `rabbit-mq-request-credentials`

Request RabbitMQ credentials for a certain role.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the role.
`rabbitmq_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

GET /{rabbitmq_mount_path}/roles

Operation ID: `rabbit-mq-list-roles`

Manage the roles that can be created with this backend.

Parameters

Parameter	Type	Location	Required	Description
`rabbitmq_mount_path`	string	path	yes	Path that the backend was mounted at
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /{rabbitmq_mount_path}/roles/{name}

Operation ID: `rabbit-mq-read-role`

Manage the roles that can be created with this backend.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the role.
`rabbitmq_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /{rabbitmq_mount_path}/roles/{name}

Operation ID: `rabbit-mq-write-role`

Manage the roles that can be created with this backend.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the role.
`rabbitmq_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`tags`	string	no	Comma-separated list of tags for this role.
`vhost_topics`	string	no	A nested map of virtual hosts and exchanges to topic permissions.
`vhosts`	string	no	A map of virtual hosts to permissions.

Responses

200: OK

DELETE /{rabbitmq_mount_path}/roles/{name}

Operation ID: `rabbit-mq-delete-role`

Manage the roles that can be created with this backend.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the role.
`rabbitmq_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: empty body

GET /{ssh_mount_path}/config/ca

Operation ID: `ssh-read-ca-configuration`

Set the SSH private key used for signing certificates.

Parameters

Parameter	Type	Location	Required	Description
`ssh_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /{ssh_mount_path}/config/ca

Operation ID: `ssh-configure-ca`

Set the SSH private key used for signing certificates.

Parameters

Parameter	Type	Location	Required	Description
`ssh_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`generate_signing_key`	boolean (default: True)	no	Generate SSH key pair internally rather than use the private_key and public_key fields.
`key_bits`	integer (default: 0)	no	Specifies the desired key bits when generating variable-length keys (such as when key_type="ssh-rsa") or which NIST P-curve to use when key_type="ec" (256, 384, or 521).
`key_type`	string (default: ssh-rsa)	no	Specifies the desired key type when generating; could be a OpenSSH key type identifier (ssh-rsa, ecdsa-sha2-nistp256, ecdsa-sha2-nistp384, ecdsa-sha2-nistp521, or ssh-ed25519) or an algorithm (rsa, ec, ed25519).
`managed_key_id`	string	no	The id of the managed key to use. When using a managed key, this field or managed_key_name is required.
`managed_key_name`	string	no	The name of the managed key to use. When using a managed key, this field or managed_key_id is required.
`private_key`	string	no	Private half of the SSH key that will be used to sign certificates.
`public_key`	string	no	Public half of the SSH key that will be used to sign certificates.

Responses

200: OK

DELETE /{ssh_mount_path}/config/ca

Operation ID: `ssh-delete-ca-configuration`

Set the SSH private key used for signing certificates.

Parameters

Parameter	Type	Location	Required	Description
`ssh_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: empty body

GET /{ssh_mount_path}/config/zeroaddress

Operation ID: `ssh-read-zero-address-configuration`

Assign zero address as default CIDR block for select roles.

Parameters

Parameter	Type	Location	Required	Description
`ssh_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /{ssh_mount_path}/config/zeroaddress

Operation ID: `ssh-configure-zero-address`

Assign zero address as default CIDR block for select roles.

Parameters

Parameter	Type	Location	Required	Description
`ssh_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`roles`	array	no	[Required] Comma separated list of role names which allows credentials to be requested for any IP address. CIDR blocks previously registered under these roles will be ignored.

Responses

200: OK

DELETE /{ssh_mount_path}/config/zeroaddress

Operation ID: `ssh-delete-zero-address-configuration`

Assign zero address as default CIDR block for select roles.

Parameters

Parameter	Type	Location	Required	Description
`ssh_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: empty body

POST /{ssh_mount_path}/creds/{role}

Operation ID: `ssh-generate-credentials`

Creates a credential for establishing SSH connection with the remote host.

Parameters

Parameter	Type	Location	Required	Description
`role`	string	path	yes	[Required] Name of the role
`ssh_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`ip`	string	no	[Required] IP of the remote host
`username`	string	no	[Optional] Username in remote host

Responses

200: OK

POST /{ssh_mount_path}/issue/{role}

Operation ID: `ssh-issue-certificate`

Request a certificate using a certain role with the provided details.

Parameters

Parameter	Type	Location	Required	Description
`role`	string	path	yes	The desired role with configuration for this request.
`ssh_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`cert_type`	string (default: user)	no	Type of certificate to be created; either “user” or “host”.
`critical_options`	object	no	Critical options that the certificate should be signed for.
`extensions`	object	no	Extensions that the certificate should be signed for.
`key_bits`	integer (default: 0)	no	Specifies the number of bits to use for the generated keys.
`key_id`	string	no	Key id that the created certificate should have. If not specified, the display name of the token will be used.
`key_type`	string (default: rsa)	no	Specifies the desired key type; must be `rsa`, `ed25519` or `ec`
`ttl`	integer	no	The requested Time To Live for the SSH certificate; sets the expiration date. If not specified the role default, backend default, or system default TTL is used, in that order. Cannot be later than the role max TTL.
`valid_principals`	string	no	Valid principals, either usernames or hostnames, that the certificate should be signed for.

Responses

200: OK

POST /{ssh_mount_path}/lookup

Operation ID: `ssh-list-roles-by-ip`

List all the roles associated with the given IP address.

Parameters

Parameter	Type	Location	Required	Description
`ssh_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`ip`	string	no	[Required] IP address of remote host

Responses

200: OK

GET /{ssh_mount_path}/public_key

Operation ID: `ssh-read-public-key`

Retrieve the public key.

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`ssh_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

GET /{ssh_mount_path}/roles

Operation ID: `ssh-list-roles`

Manage the ‘roles’ that can be created with this backend.

Parameters

Parameter	Type	Location	Required	Description
`ssh_mount_path`	string	path	yes	Path that the backend was mounted at
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /{ssh_mount_path}/roles/{role}

Operation ID: `ssh-read-role`

Manage the ‘roles’ that can be created with this backend.

Parameters

Parameter	Type	Location	Required	Description
`role`	string	path	yes	[Required for all types] Name of the role being created.
`ssh_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /{ssh_mount_path}/roles/{role}

Operation ID: `ssh-write-role`

Manage the ‘roles’ that can be created with this backend.

Parameters

Parameter	Type	Location	Required	Description
`role`	string	path	yes	[Required for all types] Name of the role being created.
`ssh_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`algorithm_signer`	string (, default, ssh-rsa, rsa-sha2-256, rsa-sha2-512)	no	[Not applicable for OTP type] [Optional for CA type] When supplied, this value specifies a signing algorithm for the key. Possible values: ssh-rsa, rsa-sha2-256, rsa-sha2-512, default, or the empty string.
`allow_bare_domains`	boolean	no	[Not applicable for OTP type] [Optional for CA type] If set, host certificates that are requested are allowed to use the base domains listed in “allowed_domains”, e.g. “example.com”. This is a separate option as in some cases this can be considered a security threat.
`allow_empty_principals`	boolean	no	[Optional for CA type] If true, host and user certificates can be issued without any valid principals. For host certificates, this means that any domain a host claims to be will be trusted by the connecting client. For user certificates, when a CA certificate is placed in a user’s AuthorizedKeys file, any principal on that certificate will be allowed to connect. When allowed_users or allowed_domains is set to * (corresponding to the role/certificate type), allow_empty_principals=false still permits issuance. It is recommend to leave this disabled.
`allow_host_certificates`	boolean (default: False)	no	[Not applicable for OTP type] [Optional for CA type] If set, certificates are allowed to be signed for use as a ‘host’.
`allow_subdomains`	boolean	no	[Not applicable for OTP type] [Optional for CA type] If set, host certificates that are requested are

Parameter	Type	Required	Description
			allowed to use subdomains of those listed in “allowed_domains”.
`allow_user_certificates`	boolean (default: False)	no	[Not applicable for OTP type] [Optional for CA type] If set, certificates are allowed to be signed for use as a ‘user’.
`allow_user_key_ids`	boolean	no	[Not applicable for OTP type] [Optional for CA type] If true, users can override the key ID for a signed certificate with the “key_id” field. When false, the key ID will always be the token display name. The key ID is logged by the SSH server and can be useful for auditing.
`allowed_critical_options`	string	no	[Not applicable for OTP type] [Optional for CA type] A comma-separated list of critical options that certificates can have when signed. To allow any critical options, set this to an empty string.
`allowed_domains`	string	no	[Not applicable for OTP type] [Optional for CA type] If this option is not specified, client can request for a signed certificate for any valid host. If only certain domains are allowed, then this list enforces it.
`allowed_domains_template`	boolean (default: False)	no	[Not applicable for OTP type] [Optional for CA type] If set, Allowed domains can be specified using identity template policies. Non-templated domains are also permitted.
`allowed_extensions`	string	no	[Not applicable for OTP type] [Optional for CA type] A comma-separated list of extensions that certificates can have when signed. An empty list means that no extension

Parameter	Type	Required	Description
			overrides are allowed by an end-user; explicitly specify '*' to allow any extensions to be set.
`allowed_user_key_lengths`	object	no	[Not applicable for OTP type] [Optional for CA type] If set, allows the enforcement of key types and minimum key sizes to be signed.
`allowed_users`	string	no	[Optional for all types] [Works differently for CA type] If this option is not specified, or is '*', client can request a credential for any valid user at the remote host, including the admin user. If only certain usernames are to be allowed, then this list enforces it. If this field is set, then credentials can only be created for default_user and usernames present in this list. Setting this option will enable all the users with access to this role to fetch credentials for all other usernames in this list. Use with caution. N.B.: with the CA type, an empty list means that no users are allowed; explicitly specify '' to allow any user.
`allowed_users_template`	boolean (default: False)	no	[Not applicable for OTP type] [Optional for CA type] If set, Allowed users can be specified using identity template policies. Non-templated users are also permitted.
`cidr_list`	string	no	[Optional for OTP type] [Not applicable for CA type] Comma separated list of CIDR blocks for which the role is applicable for. CIDR blocks can belong to more than one role.
`default_critical_options`	object	no	

Parameter	Type	Required	Description
			[Not applicable for OTP type] [Optional for CA type] Critical options certificates should have if none are provided when signing. This field takes in key value pairs in JSON format. Note that these are not restricted by “allowed_critical_options”. Defaults to none.
`default_extensions`	object	no	[Not applicable for OTP type] [Optional for CA type] Extensions certificates should have if none are provided when signing. This field takes in key value pairs in JSON format. Note that these are not restricted by “allowed_extensions”. Defaults to none.
`default_extensions_template`	boolean (default: False)	no	[Not applicable for OTP type] [Optional for CA type] If set, Default extension values can be specified using identity template policies. Non-templated extension values are also permitted.
`default_user`	string	no	[Required for OTP type] [Optional for CA type] Default username for which a credential will be generated. When the endpoint ‘creds/’ is used without a username, this value will be used as default username.
`default_user_template`	boolean (default: False)	no	[Not applicable for OTP type] [Optional for CA type] If set, Default user can be specified using identity template policies. Non-templated users are also permitted.
`exclude_cidr_list`	string	no	[Optional for OTP type] [Not applicable for CA type] Comma separated list of CIDR blocks. IP

Parameter	Type	Required	Description
			addresses belonging to these blocks are not accepted by the role. This is particularly useful when big CIDR blocks are being used by the role and certain parts of it needs to be kept out.
`key_id_format`	string	no	[Not applicable for OTP type] [Optional for CA type] When supplied, this value specifies a custom format for the key id of a signed certificate. The following variables are available for use: '{{token_display_name}}' - The display name of the token used to make the request. '{{role_name}}' - The name of the role signing the request. '{{public_key_hash}}' - A SHA256 checksum of the public key that is being signed.
`key_type`	string (otp, ca)	no	[Required for all types] Type of key used to login to hosts. It can be either 'otp' or 'ca'. 'otp' type requires agent to be installed in remote hosts.
`max_ttl`	integer	no	[Not applicable for OTP type] [Optional for CA type] The maximum allowed lease duration
`not_before_duration`	integer (default: 30)	no	[Not applicable for OTP type] [Optional for CA type] The duration that the SSH certificate should be backdated by at issuance.
`port`	integer	no	[Optional for OTP type] [Not applicable for CA type] Port number for SSH connection. Default is '22'. Port number does not play any role in creation of OTP. For 'otp' type, this is just a way to inform client about the port number to use. Port number will

Parameter	Type	Required	Description
			be returned to client by Vault server along with OTP.
`ttl`	integer	no	[Not applicable for OTP type] [Optional for CA type] The lease duration if no specific lease duration is requested. The lease duration controls the expiration of certificates issued by this backend. Defaults to the value of max_ttl.

Responses

200: OK

DELETE /{ssh_mount_path}/roles/{role}

Operation ID: `ssh-delete-role`

Manage the ‘roles’ that can be created with this backend.

Parameters

Parameter	Type	Location	Required	Description
`role`	string	path	yes	[Required for all types] Name of the role being created.
`ssh_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: empty body

POST /{ssh_mount_path}/sign/{role}

Operation ID: `ssh-sign-certificate`

Request signing an SSH key using a certain role with the provided details.

Parameters

Parameter	Type	Location	Required	Description
`role`	string	path	yes	The desired role with configuration for this request.
`ssh_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`cert_type`	string (default: user)	no	Type of certificate to be created; either “user” or “host”.
`critical_options`	object	no	Critical options that the certificate should be signed for.
`extensions`	object	no	Extensions that the certificate should be signed for.
`key_id`	string	no	Key id that the created certificate should have. If not specified, the display name of the token will be used.
`public_key`	string	no	SSH public key that should be signed.
`ttl`	integer	no	The requested Time To Live for the SSH certificate; sets the expiration date. If not specified the role default, backend default, or system default TTL is used, in that order. Cannot be later than the role max TTL.
`valid_principals`	string	no	Valid principals, either usernames or hostnames, that the certificate should be signed for.

Responses

200: OK

DELETE /{ssh_mount_path}/tidy/dynamic-keys

Operation ID: `ssh-tidy-dynamic-host-keys`

This endpoint removes the stored host keys used for the removed Dynamic Key feature, if present.

Parameters

Parameter	Type	Location	Required	Description
`ssh_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: empty body

POST /{ssh_mount_path}/verify

Operation ID: `ssh-verify-otp`

Validate the OTP provided by Vault SSH Agent.

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`ssh_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`otp`	string	no	[Required] One-Time-Key that needs to be validated

Responses

200: OK

GET /{totp_mount_path}/code/{name}

Operation ID: `totp-generate-code`

Request time-based one-time use password or validate a password for a certain key .

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the key.
`totp_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /{totp_mount_path}/code/{name}

Operation ID: `totp-validate-code`

Request time-based one-time use password or validate a password for a certain key .

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the key.
`totp_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`code`	string	no	TOTP code to be validated.

Responses

200: OK

GET /{totp_mount_path}/keys

Operation ID: `totp-list-keys`

Manage the keys that can be created with this backend.

Parameters

Parameter	Type	Location	Required	Description
`totp_mount_path`	string	path	yes	Path that the backend was mounted at
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /{totp_mount_path}/keys/{name}

Operation ID: `totp-read-key`

Manage the keys that can be created with this backend.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the key.
`totp_mount_path`	string	path	yes	Path that the backend was mounted at
`list`	string	query	no	Return a list if `true`

Responses

200: OK

POST /{totp_mount_path}/keys/{name}

Operation ID: `totp-create-key`

Manage the keys that can be created with this backend.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the key.
`totp_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`account_name`	string	no	The name of the account associated with the key. Required if generate is true.
`algorithm`	string (default: SHA1)	no	The hashing algorithm used to generate the TOTP token. Options include SHA1, SHA256 and SHA512.
`digits`	integer (default: 6)	no	The number of digits in the generated TOTP token. This value can either be 6 or 8.
`exported`	boolean (default: True)	no	Determines if a QR code and url are returned upon generating a key. Only used if generate is true.
`generate`	boolean (default: False)	no	Determines if a key should be generated by Vault or if a key is being passed from another service.
`issuer`	string	no	The name of the key’s issuing organization. Required if generate is true.
`key`	string	no	The shared master key used to generate a TOTP token. Only used if generate is false.
`key_size`	integer (default: 20)	no	Determines the size in bytes of the generated key. Only used if generate is true.
`period`	integer (default: 30)	no	The length of time used to generate a counter for the TOTP token calculation.
`qr_size`	integer (default: 200)	no	The pixel size of the generated square QR code. Only used if generate is true and exported is true. If this value is 0, a QR code will not be returned.
`skew`	integer (default: 1)	no	The number of delay periods that are allowed when validating a TOTP token. This value can either be 0 or 1. Only used if generate is true.
`url`	string	no	A TOTP url string containing all of the parameters for key setup. Only used if generate is false.

Responses

200: OK

DELETE /{totp_mount_path}/keys/{name}

Operation ID: `totp-delete-key`

Manage the keys that can be created with this backend.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the key.
`totp_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: empty body

GET /{transit_mount_path}/backup/{name}

Operation ID: `transit-back-up-key`

Backup the named key

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the key
`transit_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

GET /{transit_mount_path}/byok-export/{destination}/{source}

Operation ID: `transit-byok-key`

Securely export named encryption or signing key

Parameters

Parameter	Type	Location	Required	Description
`destination`	string	path	yes	Destination key to export to; usually the public wrapping key of another Transit instance.
`source`	string	path	yes	Source key to export; could be any present key within Transit.
`transit_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

GET /{transit_mount_path}/byok-export/{destination}/{source}/{version}

Operation ID: `transit-byok-key-version`

Securely export named encryption or signing key

Parameters

Parameter	Type	Location	Required	Description
`destination`	string	path	yes	Destination key to export to; usually the public wrapping key of another Transit instance.
`source`	string	path	yes	Source key to export; could be any present key within Transit.
`version`	string	path	yes	Optional version of the key to export, else all key versions are exported.
`transit_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

GET /{transit_mount_path}/cache-config

Operation ID: `transit-read-cache-configuration`

Returns the size of the active cache

Parameters

Parameter	Type	Location	Required	Description
`transit_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /{transit_mount_path}/cache-config

Operation ID: `transit-configure-cache`

Configures a new cache of the specified size

Creation supported: yes

Parameters

Parameter	Type	Location	Required	Description
`transit_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`size`	integer (default: 0)	no	Size of cache, use 0 for an unlimited cache size, defaults to 0

Responses

200: OK

GET /{transit_mount_path}/config/keys

Operation ID: `transit-read-keys-configuration`

Configuration common across all keys

Parameters

Parameter	Type	Location	Required	Description
`transit_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /{transit_mount_path}/config/keys

Operation ID: `transit-configure-keys`

Configuration common across all keys

Parameters

Parameter	Type	Location	Required	Description
`transit_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`disable_upsert`	boolean	no	Whether to allow automatic upserting (creation) of keys on the encrypt endpoint.

Responses

200: OK

POST /{transit_mount_path}/datakey/{plaintext}/{name}

Operation ID: `transit-generate-data-key`

Generate a data key

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	The backend key used for encrypting the data key
`plaintext`	string	path	yes	“plaintext” will return the key in both plaintext and ciphertext; “wrapped” will return the ciphertext only.
`transit_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`bits`	integer (default: 256)	no	Number of bits for the key; currently 128, 256, and 512 bits are supported. Defaults to 256.
`context`	string	no	Context for key derivation. Required for derived keys.
`key_version`	integer	no	The version of the Vault key to use for encryption of the data key. Must be 0 (for latest) or a value greater than or equal to the min_encryption_version configured on the key.
`nonce`	string	no	Nonce for when convergent encryption v1 is used (only in Vault 0.6.1)

Responses

200: OK

POST /{transit_mount_path}/decrypt/{name}

Operation ID: `transit-decrypt`

Decrypt a ciphertext value using a named key

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the key
`transit_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`associated_data`	string	no	When using an AEAD cipher mode, such as AES-GCM, this parameter allows passing associated data (AD/AAD) into the encryption function; this data must be passed on subsequent decryption requests but can be transited in plaintext. On successful decryption, both the ciphertext and the associated data are attested not to have been tampered with.
`batch_input`	array	no	Specifies a list of items to be decrypted in a single batch. When this parameter is set, if the parameters ‘ciphertext’, ‘context’ and ’nonce’ are also set, they will be ignored. Any batch output will preserve the order of the batch input.
`ciphertext`	string	no	The ciphertext to decrypt, provided as returned by encrypt.
`context`	string	no	Base64 encoded context for key derivation. Required if key derivation is enabled.
`nonce`	string	no	Base64 encoded nonce value used during encryption. Must be provided if convergent encryption is enabled for this key and the key was generated with Vault 0.6.1. Not required for keys created in 0.6.2+.
`partial_failure_response_code`	integer	no	Ordinarily, if a batch item fails to decrypt due to a bad input, but other batch items succeed, the HTTP response code is 400 (Bad Request). Some applications may want to treat partial failures differently. Providing the parameter returns the given response code integer instead of a 400 in this case. If all values fail HTTP 400 is still returned.

Responses

200: OK

POST /{transit_mount_path}/encrypt/{name}

Operation ID: `transit-encrypt`

Encrypt a plaintext value or a batch of plaintext blocks using a named key

Creation supported: yes

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the key
`transit_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`associated_data`	string	no	When using an AEAD cipher mode, such as AES-GCM, this parameter allows passing associated data (AD/ AAD) into the encryption function; this data must be passed on subsequent decryption requests but can be transited in plaintext. On successful decryption, both the ciphertext and the associated data are attested not to have been tampered with.
`batch_input`	array	no	Specifies a list of items to be encrypted in a single batch. When this parameter is set, if the parameters ‘plaintext’, ‘context’ and ’nonce’ are also set, they will be ignored. Any batch output will preserve the order of the batch input.
`context`	string	no	Base64 encoded context for key derivation. Required if key derivation is enabled
`convergent_encryption`	boolean	no	This parameter will only be used when a key is expected to be created. Whether to support convergent encryption. This is only supported when using a key with key derivation enabled and will require all requests to carry both a context and 96-bit (12-byte) nonce. The given nonce will be used in place of a randomly generated nonce. As a result, when the same context and nonce are supplied, the same ciphertext is generated. It is *very important* when using this mode that you ensure that all nonces are unique for a given context. Failing to do so will severely impact the ciphertext’s security.

Parameter	Type	Required	Description
`key_version`	integer	no	The version of the key to use for encryption. Must be 0 (for latest) or a value greater than or equal to the min_encryption_version configured on the key.
`nonce`	string	no	Base64 encoded nonce value. Must be provided if convergent encryption is enabled for this key and the key was generated with Vault 0.6.1. Not required for keys created in 0.6.2+. The value must be exactly 96 bits (12 bytes) long and the user must ensure that for any given context (and thus, any given encryption key) this nonce value is **never reused**.
`partial_failure_response_code`	integer	no	Ordinarily, if a batch item fails to encrypt due to a bad input, but other batch items succeed, the HTTP response code is 400 (Bad Request). Some applications may want to treat partial failures differently. Providing the parameter returns the given response code integer instead of a 400 in this case. If all values fail HTTP 400 is still returned.
`plaintext`	string	no	Base64 encoded plaintext value to be encrypted
`type`	string (default: aes256-gcm96)	no	This parameter is required when encryption key is expected to be created. When performing an upsert operation, the type of key to create. Currently, “aes128-gcm96” (symmetric) and “aes256-gcm96” (symmetric) are the only types supported. Defaults to “aes256-gcm96”.

Responses

200: OK

GET /{transit_mount_path}/export/{type}/{name}

Operation ID: `transit-export-key`

Export named encryption or signing key

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the key
`type`	string	path	yes	Type of key to export (encryption-key, signing-key, hmac-key, public-key)
`transit_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

GET /{transit_mount_path}/export/{type}/{name}/{version}

Operation ID: `transit-export-key-version`

Export named encryption or signing key

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the key
`type`	string	path	yes	Type of key to export (encryption-key, signing-key, hmac-key, public-key)
`version`	string	path	yes	Version of the key
`transit_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /{transit_mount_path}/hash

Operation ID: `transit-hash`

Generate a hash sum for input data

Parameters

Parameter	Type	Location	Required	Description
`transit_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`algorithm`	string (default: sha2-256)	no	Algorithm to use (POST body parameter). Valid values are: *sha2-224* sha2-256 *sha2-384* sha2-512 *sha3-224* sha3-256 *sha3-384* sha3-512 *streebog-256* streebog-512 Defaults to “sha2-256”.
`format`	string (default: hex)	no	Encoding format to use. Can be “hex” or “base64”. Defaults to “hex”.
`input`	string	no	The base64-encoded input data
`urlalgorithm`	string	no	Algorithm to use (POST URL parameter)

Responses

200: OK

POST /{transit_mount_path}/hash/{urlalgorithm}

Operation ID: `transit-hash-with-algorithm`

Generate a hash sum for input data

Parameters

Parameter	Type	Location	Required	Description
`urlalgorithm`	string	path	yes	Algorithm to use (POST URL parameter)
`transit_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`algorithm`	string (default: sha2-256)	no	Algorithm to use (POST body parameter). Valid values are: *sha2-224* sha2-256 *sha2-384* sha2-512 *sha3-224* sha3-256 *sha3-384* sha3-512 *streebog-256* streebog-512 Defaults to “sha2-256”.
`format`	string (default: hex)	no	Encoding format to use. Can be “hex” or “base64”. Defaults to “hex”.
`input`	string	no	The base64-encoded input data

Responses

200: OK

POST /{transit_mount_path}/hmac/{name}

Operation ID: `transit-generate-hmac`

Generate an HMAC for input data using the named key

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	The key to use for the HMAC function
`transit_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`algorithm`	string (default: sha2-256)	no	Algorithm to use (POST body parameter). Valid values are: *sha2-224* sha2-256 *sha2-384* sha2-512 *sha3-224* sha3-256 *sha3-384* sha3-512 *streebog-256* streebog-512 Defaults to “sha2-256”.
`batch_input`	array	no	Specifies a list of items to be processed in a single batch. When this parameter is set, if the parameter ‘input’ is also set, it will be ignored. Any batch output will preserve the order of the batch input.
`input`	string	no	The base64-encoded input data
`key_version`	integer	no	The version of the key to use for generating the HMAC. Must be 0 (for latest) or a value greater than or equal to the min_encryption_version configured on the key.
`urlalgorithm`	string	no	Algorithm to use (POST URL parameter)

Responses

200: OK

POST /{transit_mount_path}/hmac/{name}/{urlalgorithm}

Operation ID: `transit-generate-hmac-with-algorithm`

Generate an HMAC for input data using the named key

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	The key to use for the HMAC function
`urlalgorithm`	string	path	yes	Algorithm to use (POST URL parameter)
`transit_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`algorithm`	string (default: sha2-256)	no	Algorithm to use (POST body parameter). Valid values are: *sha2-224* sha2-256 *sha2-384* sha2-512 *sha3-224* sha3-256 *sha3-384* sha3-512 *streebog-256* streebog-512 Defaults to “sha2-256”.
`batch_input`	array	no	Specifies a list of items to be processed in a single batch. When this parameter is set, if the parameter ‘input’ is also set, it will be ignored. Any batch output will preserve the order of the batch input.
`input`	string	no	The base64-encoded input data
`key_version`	integer	no	The version of the key to use for generating the HMAC. Must be 0 (for latest) or a value greater than or equal to the min_encryption_version configured on the key.

Responses

200: OK

GET /{transit_mount_path}/keys

Operation ID: `transit-list-keys`

Managed named encryption keys

Parameters

Parameter	Type	Location	Required	Description
`transit_mount_path`	string	path	yes	Path that the backend was mounted at
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /{transit_mount_path}/keys/{name}

Operation ID: `transit-read-key`

Managed named encryption keys

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the key
`transit_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

POST /{transit_mount_path}/keys/{name}

Operation ID: `transit-create-key`

Managed named encryption keys

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the key
`transit_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`allow_plaintext_backup`	boolean	no	Enables taking a backup of the named key in plaintext format. Once set, this cannot be disabled.
`auto_rotate_period`	integer (default: 0)	no	Amount of time the key should live before being automatically rotated. A value of 0 (default) disables automatic rotation for the key.
`context`	string	no	Base64 encoded context for key derivation. When reading a key with key derivation enabled, if the key type supports public keys, this will return the public key for the given context.
`convergent_encryption`	boolean	no	Whether to support convergent encryption. This is only supported when using a key with key derivation enabled and will require all requests to carry both a context and 96-bit (12-byte) nonce. The given nonce will be used in place of a randomly generated nonce. As a result, when the same context and nonce are supplied, the same ciphertext is generated. It is *very important* when using this mode that you ensure that all nonces are unique for a given context. Failing to do so will severely impact the ciphertext’s security.
`derived`	boolean	no	Enables key derivation mode. This allows for per-transaction unique keys for encryption operations.
`exportable`	boolean	no	Enables keys to be exportable. This allows for all the valid keys in the key ring to be exported.
`key_size`	integer (default: 0)	no	The key size in bytes for the algorithm. Only applies to HMAC and must be no fewer than 32 bytes and no more than 512
`managed_key_id`	string	no	The UUID of the managed key to use for this transit key

Parameter	Type	Required	Description
`managed_key_name`	string	no	The name of the managed key to use for this transit key
`type`	string (default: aes256-gcm96)	no	The type of key to create. Currently, “aes128-gcm96” (symmetric), “aes256-gcm96” (symmetric), “ecdsa-p256” (asymmetric), “ecdsa-p384” (asymmetric), “ecdsa-p521” (asymmetric), “ed25519” (asymmetric), “rsa-2048” (asymmetric), “rsa-3072” (asymmetric), “rsa-4096” (asymmetric), “gost28147” (symmetric), “gost3410-256-paramset-a”, “gost3410-256-paramset-b”, “gost3410-256-paramset-c”, “gost3410-256-paramset-d”, “gost3410-512-paramset-a”, “gost3410-512-paramset-b”, “gost3410-512-paramset-c”, “gost341264” (symmetric), “gost3412128” (symmetric) are supported. Defaults to “aes256-gcm96”.

Responses

200: OK

DELETE /{transit_mount_path}/keys/{name}

Operation ID: `transit-delete-key`

Managed named encryption keys

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the key
`transit_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

204: empty body

POST /{transit_mount_path}/keys/{name}/config

Operation ID: `transit-configure-key`

Configure a named encryption key

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the key
`transit_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`allow_plaintext_backup`	boolean	no	Enables taking a backup of the named key in plaintext format. Once set, this cannot be disabled.
`auto_rotate_period`	integer	no	Amount of time the key should live before being automatically rotated. A value of 0 disables automatic rotation for the key.
`deletion_allowed`	boolean	no	Whether to allow deletion of the key
`exportable`	boolean	no	Enables export of the key. Once set, this cannot be disabled.
`min_decryption_version`	integer	no	If set, the minimum version of the key allowed to be decrypted. For signing keys, the minimum version allowed to be used for verification.
`min_encryption_version`	integer	no	If set, the minimum version of the key allowed to be used for encryption; or for signing keys, to be used for signing. If set to zero, only the latest version of the key is allowed.

Responses

200: OK

POST /{transit_mount_path}/keys/{name}/import

Operation ID: `transit-import-key`

Imports an externally-generated key into a new transit key

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	The name of the key
`transit_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`allow_plaintext_backup`	boolean	no	Enables taking a backup of the named key in plaintext format. Once set, this cannot be disabled.
`allow_rotation`	boolean	no	True if the imported key may be rotated within Vault; false otherwise.
`auto_rotate_period`	integer (default: 0)	no	Amount of time the key should live before being automatically rotated. A value of 0 (default) disables automatic rotation for the key.
`ciphertext`	string	no	The base64-encoded ciphertext of the keys. The AES key should be encrypted using OAEP with the wrapping key and then concatenated with the import key, wrapped by the AES key.
`context`	string	no	Base64 encoded context for key derivation. When reading a key with key derivation enabled, if the key type supports public keys, this will return the public key for the given context.
`derived`	boolean	no	Enables key derivation mode. This allows for per-transaction unique keys for encryption operations.
`exportable`	boolean	no	Enables keys to be exportable. This allows for all the valid keys in the key ring to be exported.
`hash_function`	string (default: SHA256)	no	The hash function used as a random oracle in the OAEP wrapping of the user-generated, ephemeral AES key. Can be one of “SHA1”, “SHA224”, “SHA256” (default), “SHA384”, or “SHA512”
`public_key`	string	no	The plaintext PEM public key to be imported. If “ciphertext” is set, this field is ignored.
`type`	string (default:	no	The type of key being imported. Currently, “aes128-gcm96” (symmetric), “aes256-gcm96” (symmetric), “ecdsa-p256”

Parameter	Type	Required	Description
	aes256-gcm96)		(asymmetric), “ecdsa-p384” (asymmetric), “ecdsa-p521” (asymmetric), “ed25519” (asymmetric), “rsa-2048” (asymmetric), “rsa-3072” (asymmetric), “rsa-4096” (asymmetric), “gost28147” (symmetric), “gost3410-256-paramset-a”, “gost3410-256-paramset-b”, “gost3410-256-paramset-c”, “gost3410-256-paramset-d”, “gost3410-512-paramset-a”, “gost3410-512-paramset-b”, “gost3410-512-paramset-c”, “gost341264” (symmetric), “gost3412128” (symmetric) are supported. Defaults to “aes256-gcm96”.

Responses

200: OK

POST /{transit_mount_path}/keys/{name}/import_version

Operation ID: `transit-import-key-version`

Imports an externally-generated key into an existing imported key

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	The name of the key
`transit_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`ciphertext`	string	no	The base64-encoded ciphertext of the keys. The AES key should be encrypted using OAEP with the wrapping key and then concatenated with the import key, wrapped by the AES key.
`hash_function`	string (default: SHA256)	no	The hash function used as a random oracle in the OAEP wrapping of the user-generated, ephemeral AES key. Can be one of “SHA1”, “SHA224”, “SHA256” (default), “SHA384”, or “SHA512”
`public_key`	string	no	The plaintext public key to be imported. If “ciphertext” is set, this field is ignored.
`version`	integer	no	Key version to be updated, if left empty, a new version will be created unless a private key is specified and the ‘Latest’ key is missing a private key.

Responses

200: OK

POST /{transit_mount_path}/keys/{name}/rotate

Operation ID: `transit-rotate-key`

Rotate named encryption key

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the key
`transit_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`managed_key_id`	string	no	The UUID of the managed key to use for the new version of this transit key
`managed_key_name`	string	no	The name of the managed key to use for the new version of this transit key

Responses

200: OK

POST /{transit_mount_path}/keys/{name}/trim

Operation ID: `transit-trim-key`

Trim key versions of a named key

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the key
`transit_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`min_available_version`	integer	no	The minimum available version for the key ring. All versions before this version will be permanently deleted. This value can at most be equal to the lesser of ‘min_decryption_version’ and ‘min_encryption_version’. This is not allowed to be set when either ‘min_encryption_version’ or ‘min_decryption_version’ is set to zero.

Responses

200: OK

POST /{transit_mount_path}/random

Operation ID: `transit-generate-random`

Generate random bytes

Parameters

Parameter	Type	Location	Required	Description
`transit_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`bytes`	integer (default: 32)	no	The number of bytes to generate (POST body parameter). Defaults to 32 (256 bits).
`format`	string (default: base64)	no	Encoding format to use. Can be “hex” or “base64”. Defaults to “base64”.
`source`	string (default: platform)	no	Which system to source random data from, ether “platform”, “seal”, or “all”.
`urlbytes`	string	no	The number of bytes to generate (POST URL parameter)

Responses

200: OK

POST /{transit_mount_path}/random/{source}

Operation ID: `transit-generate-random-with-source`

Generate random bytes

Parameters

Parameter	Type	Location	Required	Description
`source`	string	path	yes	Which system to source random data from, ether “platform”, “seal”, or “all”.
`transit_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`bytes`	integer (default: 32)	no	The number of bytes to generate (POST body parameter). Defaults to 32 (256 bits).
`format`	string (default: base64)	no	Encoding format to use. Can be “hex” or “base64”. Defaults to “base64”.
`urlbytes`	string	no	The number of bytes to generate (POST URL parameter)

Responses

200: OK

POST /{transit_mount_path}/random/{source}/{urlbytes}

Operation ID: `transit-generate-random-with-source-and-bytes`

Generate random bytes

Parameters

Parameter	Type	Location	Required	Description
`source`	string	path	yes	Which system to source random data from, ether “platform”, “seal”, or “all”.
`urlbytes`	string	path	yes	The number of bytes to generate (POST URL parameter)
`transit_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`bytes`	integer (default: 32)	no	The number of bytes to generate (POST body parameter). Defaults to 32 (256 bits).
`format`	string (default: base64)	no	Encoding format to use. Can be “hex” or “base64”. Defaults to “base64”.

Responses

200: OK

POST /{transit_mount_path}/random/{urlbytes}

Operation ID: `transit-generate-random-with-bytes`

Generate random bytes

Parameters

Parameter	Type	Location	Required	Description
`urlbytes`	string	path	yes	The number of bytes to generate (POST URL parameter)
`transit_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`bytes`	integer (default: 32)	no	The number of bytes to generate (POST body parameter). Defaults to 32 (256 bits).
`format`	string (default: base64)	no	Encoding format to use. Can be “hex” or “base64”. Defaults to “base64”.
`source`	string (default: platform)	no	Which system to source random data from, ether “platform”, “seal”, or “all”.

Responses

200: OK

POST /{transit_mount_path}/restore

Operation ID: `transit-restore-key`

Restore the named key

Parameters

Parameter	Type	Location	Required	Description
`transit_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`backup`	string	no	Backed up key data to be restored. This should be the output from the ‘backup/’ endpoint.
`force`	boolean (default: False)	no	If set and a key by the given name exists, force the restore operation and override the key.
`name`	string	no	If set, this will be the name of the restored key.

Responses

200: OK

POST /{transit_mount_path}/restore/{name}

Operation ID: `transit-restore-and-rename-key`

Restore the named key

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	If set, this will be the name of the restored key.
`transit_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`backup`	string	no	Backed up key data to be restored. This should be the output from the ‘backup/’ endpoint.
`force`	boolean (default: False)	no	If set and a key by the given name exists, force the restore operation and override the key.

Responses

200: OK

POST /{transit_mount_path}/rewrap/{name}

Operation ID: `transit-rewrap`

Rewrap ciphertext

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the key
`transit_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`batch_input`	array	no	Specifies a list of items to be re-encrypted in a single batch. When this parameter is set, if the parameters ‘ciphertext’, ‘context’ and ’nonce’ are also set, they will be ignored. Any batch output will preserve the order of the batch input.
`ciphertext`	string	no	Ciphertext value to rewrap
`context`	string	no	Base64 encoded context for key derivation. Required for derived keys.
`key_version`	integer	no	The version of the key to use for encryption. Must be 0 (for latest) or a value greater than or equal to the min_encryption_version configured on the key.
`nonce`	string	no	Nonce for when convergent encryption is used

Responses

200: OK

POST /{transit_mount_path}/sign/{name}

Operation ID: `transit-sign`

Generate a signature for input data using the named key

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	The key to use
`transit_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`algorithm`	string (default: sha2-256)	no	Deprecated: use “hash_algorithm” instead.
`batch_input`	array	no	Specifies a list of items for processing. When this parameter is set, any supplied ‘input’ or ‘context’ parameters will be ignored. Responses are returned in the ‘batch_results’ array component of the ‘data’ element of the response. Any batch output will preserve the order of the batch input
`context`	string	no	Base64 encoded context for key derivation. Required if key derivation is enabled; currently only available with ed25519 keys.
`hash_algorithm`	string (default: sha2-256)	no	Hash algorithm to use (POST body parameter). Valid values are: * sha1 * sha2-224 * sha2-256 * sha2-384 * sha2-512 * sha3-224 * sha3-256 * sha3-384 * sha3-512 * none Defaults to “sha2-256”. Not valid for all key types, including ed25519. Using none requires setting prehashed=true and signature_algorithm=pkcs1v15, yielding a PKCSv1_5_NoOID instead of the usual PKCSv1_5_DERnull signature.
`input`	string	no	The base64-encoded input data
`key_version`	integer	no	The version of the key to use for signing. Must be 0 (for latest) or a value greater than or equal to the min_encryption_version configured on the key.
`marshaling_algorithm`	string (default: asn1)	no	The method by which to marshal the signature. The default is ‘asn1’ which is used by openssl and X.509. It can also be set to ‘jws’ which is used for JWT signatures; setting it to this will also cause the encoding of the signature to be url-safe base64 instead of using standard base64 encoding. Currently only valid for ECDSA P-256 key types”.

Parameter	Type	Required	Description
`prehashed`	boolean	no	Set to 'true' when the input is already hashed. If the key type is 'rsa-2048', 'rsa-3072' or 'rsa-4096', then the algorithm used to hash the input should be indicated by the 'algorithm' parameter.
`salt_length`	string (default: auto)	no	The salt length used to sign. Currently only applies to the RSA PSS signature scheme. Options are 'auto' (the default used by Golang, causing the salt to be as large as possible when signing), 'hash' (causes the salt length to equal the length of the hash used in the signature), or an integer between the minimum and the maximum permissible salt lengths for the given RSA key size. Defaults to 'auto'.
`signature_algorithm`	string	no	The signature algorithm to use for signing. Currently only applies to RSA key types. Options are 'pss' or 'pkcs1v15'. Defaults to 'pss'
`urlalgorithm`	string	no	Hash algorithm to use (POST URL parameter)

Responses

200: OK

POST /{transit_mount_path}/sign/{name}/{urlalgorithm}

Operation ID: `transit-sign-with-algorithm`

Generate a signature for input data using the named key

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	The key to use
`urlalgorithm`	string	path	yes	Hash algorithm to use (POST URL parameter)
`transit_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`algorithm`	string (default: sha2-256)	no	Deprecated: use “hash_algorithm” instead.
`batch_input`	array	no	Specifies a list of items for processing. When this parameter is set, any supplied ‘input’ or ‘context’ parameters will be ignored. Responses are returned in the ‘batch_results’ array component of the ‘data’ element of the response. Any batch output will preserve the order of the batch input
`context`	string	no	Base64 encoded context for key derivation. Required if key derivation is enabled; currently only available with ed25519 keys.
`hash_algorithm`	string (default: sha2-256)	no	Hash algorithm to use (POST body parameter). Valid values are: * sha1 * sha2-224 * sha2-256 * sha2-384 * sha2-512 * sha3-224 * sha3-256 * sha3-384 * sha3-512 * none Defaults to “sha2-256”. Not valid for all key types, including ed25519. Using none requires setting prehashed=true and signature_algorithm=pkcs1v15, yielding a PKCSv1_5_NoOID instead of the usual PKCSv1_5_DERnull signature.
`input`	string	no	The base64-encoded input data
`key_version`	integer	no	The version of the key to use for signing. Must be 0 (for latest) or a value greater than or equal to the min_encryption_version configured on the key.
`marshaling_algorithm`	string (default: asn1)	no	The method by which to marshal the signature. The default is ‘asn1’ which is used by openssl and X.509. It can also be set to ‘jws’ which is used for JWT signatures; setting it to this will also cause the encoding of the signature to be url-safe base64 instead of using standard base64 encoding. Currently only valid for ECDSA P-256 key types”.

Parameter	Type	Required	Description
`prehashed`	boolean	no	Set to 'true' when the input is already hashed. If the key type is 'rsa-2048', 'rsa-3072' or 'rsa-4096', then the algorithm used to hash the input should be indicated by the 'algorithm' parameter.
`salt_length`	string (default: auto)	no	The salt length used to sign. Currently only applies to the RSA PSS signature scheme. Options are 'auto' (the default used by Golang, causing the salt to be as large as possible when signing), 'hash' (causes the salt length to equal the length of the hash used in the signature), or an integer between the minimum and the maximum permissible salt lengths for the given RSA key size. Defaults to 'auto'.
`signature_algorithm`	string	no	The signature algorithm to use for signing. Currently only applies to RSA key types. Options are 'pss' or 'pkcs1v15'. Defaults to 'pss'

Responses

200: OK

POST /{transit_mount_path}/verify/{name}

Operation ID: `transit-verify`

Verify a signature or HMAC for input data created using the named key

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	The key to use
`transit_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`algorithm`	string (default: sha2-256)	no	Deprecated: use “hash_algorithm” instead.
`batch_input`	array	no	Specifies a list of items for processing. When this parameter is set, any supplied ‘input’, ‘hmac’ or ‘signature’ parameters will be ignored. Responses are returned in the ‘batch_results’ array component of the ‘data’ element of the response. Any batch output will preserve the order of the batch input
`context`	string	no	Base64 encoded context for key derivation. Required if key derivation is enabled; currently only available with ed25519 keys.
`hash_algorithm`	string (default: sha2-256)	no	Hash algorithm to use (POST body parameter). Valid values are: * sha1 * sha2-224 * sha2-256 * sha2-384 * sha2-512 * sha3-224 * sha3-256 * sha3-384 * sha3-512 * none Defaults to “sha2-256”. Not valid for all key types. See note about none on signing path.
`hmac`	string	no	The HMAC, including vault header/key version
`input`	string	no	The base64-encoded input data to verify
`marshaling_algorithm`	string (default: asn1)	no	The method by which to unmarshal the signature when verifying. The default is ‘asn1’ which is used by openssl and X.509; can also be set to ‘jws’ which is used for JWT signatures in which case the signature is also expected to be url-safe base64 encoding instead of standard base64 encoding. Currently only valid for ECDSA P-256 key types”.
`prehashed`	boolean	no	Set to ’true’ when the input is already hashed. If the key type is ‘rsa-2048’, ‘rsa-3072’ or ‘rsa-4096’, then the algorithm used to hash

Parameter	Type	Required	Description
			the input should be indicated by the ‘algorithm’ parameter.
`salt_length`	string (default: auto)	no	The salt length used to sign. Currently only applies to the RSA PSS signature scheme. Options are ‘auto’ (the default used by Golang, causing the salt to be as large as possible when signing), ‘hash’ (causes the salt length to equal the length of the hash used in the signature), or an integer between the minimum and the maximum permissible salt lengths for the given RSA key size. Defaults to ‘auto’.
`signature`	string	no	The signature, including vault header/key version
`signature_algorithm`	string	no	The signature algorithm to use for signature verification. Currently only applies to RSA key types. Options are ‘pss’ or ‘pkcs1v15’. Defaults to ‘pss’
`urlalgorithm`	string	no	Hash algorithm to use (POST URL parameter)

Responses

200: OK

POST /{transit_mount_path}/verify/{name}/{urlalgorithm}

Operation ID: `transit-verify-with-algorithm`

Verify a signature or HMAC for input data created using the named key

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	The key to use
`urlalgorithm`	string	path	yes	Hash algorithm to use (POST URL parameter)
`transit_mount_path`	string	path	yes	Path that the backend was mounted at

Request body parameters

Parameter	Type	Required	Description
`algorithm`	string (default: sha2-256)	no	Deprecated: use “hash_algorithm” instead.
`batch_input`	array	no	Specifies a list of items for processing. When this parameter is set, any supplied ‘input’, ‘hmac’ or ‘signature’ parameters will be ignored. Responses are returned in the ‘batch_results’ array component of the ‘data’ element of the response. Any batch output will preserve the order of the batch input
`context`	string	no	Base64 encoded context for key derivation. Required if key derivation is enabled; currently only available with ed25519 keys.
`hash_algorithm`	string (default: sha2-256)	no	Hash algorithm to use (POST body parameter). Valid values are: * sha1 * sha2-224 * sha2-256 * sha2-384 * sha2-512 * sha3-224 * sha3-256 * sha3-384 * sha3-512 * none Defaults to “sha2-256”. Not valid for all key types. See note about none on signing path.
`hmac`	string	no	The HMAC, including vault header/key version
`input`	string	no	The base64-encoded input data to verify
`marshaling_algorithm`	string (default: asn1)	no	The method by which to unmarshal the signature when verifying. The default is ‘asn1’ which is used by openssl and X.509; can also be set to ‘jws’ which is used for JWT signatures in which case the signature is also expected to be url-safe base64 encoding instead of standard base64 encoding. Currently only valid for ECDSA P-256 key types”.
`prehashed`	boolean	no	Set to ’true’ when the input is already hashed. If the key type is ‘rsa-2048’, ‘rsa-3072’ or ‘rsa-4096’, then the algorithm used to hash

Parameter	Type	Required	Description
			the input should be indicated by the ‘algorithm’ parameter.
`salt_length`	string (default: auto)	no	The salt length used to sign. Currently only applies to the RSA PSS signature scheme. Options are ‘auto’ (the default used by Golang, causing the salt to be as large as possible when signing), ‘hash’ (causes the salt length to equal the length of the hash used in the signature), or an integer between the minimum and the maximum permissible salt lengths for the given RSA key size. Defaults to ‘auto’.
`signature`	string	no	The signature, including vault header/key version
`signature_algorithm`	string	no	The signature algorithm to use for signature verification. Currently only applies to RSA key types. Options are ‘pss’ or ‘pkcs1v15’. Defaults to ‘pss’

Responses

200: OK

GET /{transit_mount_path}/wrapping_key

Operation ID: `transit-read-wrapping-key`

Returns the public key to use for wrapping imported keys

Parameters

Parameter	Type	Location	Required	Description
`transit_mount_path`	string	path	yes	Path that the backend was mounted at

Responses

200: OK

identity

identity

POST /identity/alias

Operation ID: `alias-create`

Create a new alias.

Request body parameters

Parameter	Type	Required	Description
`canonical_id`	string	no	Entity ID to which this alias belongs to
`entity_id`	string	no	Entity ID to which this alias belongs to. This field is deprecated in favor of ‘canonical_id’.
`id`	string	no	ID of the alias
`mount_accessor`	string	no	Mount accessor to which this alias belongs to
`name`	string	no	Name of the alias

Responses

200: OK

GET /identity/alias/id

Operation ID: `alias-list-by-id`

List all the alias IDs.

Parameters

Parameter	Type	Location	Required	Description
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /identity/alias/id/{id}

Operation ID: `alias-read-by-id`

Update, read or delete an alias ID.

Parameters

Parameter	Type	Location	Required	Description
`id`	string	path	yes	ID of the alias

Responses

200: OK

POST /identity/alias/id/{id}

Operation ID: `alias-update-by-id`

Update, read or delete an alias ID.

Parameters

Parameter	Type	Location	Required	Description
`id`	string	path	yes	ID of the alias

Request body parameters

Parameter	Type	Required	Description
`canonical_id`	string	no	Entity ID to which this alias should be tied to
`entity_id`	string	no	Entity ID to which this alias should be tied to. This field is deprecated in favor of ‘canonical_id’.
`mount_accessor`	string	no	Mount accessor to which this alias belongs to
`name`	string	no	Name of the alias

Responses

200: OK

DELETE /identity/alias/id/{id}

Operation ID: `alias-delete-by-id`

Update, read or delete an alias ID.

Parameters

Parameter	Type	Location	Required	Description
`id`	string	path	yes	ID of the alias

Responses

204: empty body

POST /identity/entity

Operation ID: `entity-create`

Create a new entity

Request body parameters

Parameter	Type	Required	Description
`disabled`	boolean	no	If set true, tokens tied to this identity will not be able to be used (but will not be revoked).
`id`	string	no	ID of the entity. If set, updates the corresponding existing entity.
`metadata`	object	no	Metadata to be associated with the entity. In CLI, this parameter can be repeated multiple times, and it all gets merged together. For example: stronghold metadata=key1=value1 metadata=key2=value2
`name`	string	no	Name of the entity
`policies`	array	no	Policies to be tied to the entity.

Responses

200: OK

POST /identity/entity-alias

Operation ID: `entity-create-alias`

Create a new alias.

Request body parameters

Parameter	Type	Required	Description
`canonical_id`	string	no	Entity ID to which this alias belongs
`custom_metadata`	object	no	User provided key-value pairs
`entity_id`	string	no	Entity ID to which this alias belongs. This field is deprecated, use canonical_id.
`id`	string	no	ID of the entity alias. If set, updates the corresponding entity alias.
`mount_accessor`	string	no	Mount accessor to which this alias belongs to; unused for a modify
`name`	string	no	Name of the alias; unused for a modify

Responses

200: OK

GET /identity/entity-alias/id

Operation ID: `entity-list-aliases-by-id`

List all the alias IDs.

Parameters

Parameter	Type	Location	Required	Description
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /identity/entity-alias/id/{id}

Operation ID: `entity-read-alias-by-id`

Update, read or delete an alias ID.

Parameters

Parameter	Type	Location	Required	Description
`id`	string	path	yes	ID of the alias

Responses

200: OK

POST /identity/entity-alias/id/{id}

Operation ID: `entity-update-alias-by-id`

Update, read or delete an alias ID.

Parameters

Parameter	Type	Location	Required	Description
`id`	string	path	yes	ID of the alias

Request body parameters

Parameter	Type	Required	Description
`canonical_id`	string	no	Entity ID to which this alias should be tied to
`custom_metadata`	object	no	User provided key-value pairs
`entity_id`	string	no	Entity ID to which this alias belongs to. This field is deprecated, use canonical_id.
`mount_accessor`	string	no	(Unused)
`name`	string	no	(Unused)

Responses

200: OK

DELETE /identity/entity-alias/id/{id}

Operation ID: `entity-delete-alias-by-id`

Update, read or delete an alias ID.

Parameters

Parameter	Type	Location	Required	Description
`id`	string	path	yes	ID of the alias

Responses

204: empty body

POST /identity/entity/batch-delete

Operation ID: `entity-batch-delete`

Delete all of the entities provided

Request body parameters

Parameter	Type	Required	Description
`entity_ids`	array	no	Entity IDs to delete

Responses

200: OK

GET /identity/entity/id

Operation ID: `entity-list-by-id`

List all the entity IDs

Parameters

Parameter	Type	Location	Required	Description
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /identity/entity/id/{id}

Operation ID: `entity-read-by-id`

Update, read or delete an entity using entity ID

Parameters

Parameter	Type	Location	Required	Description
`id`	string	path	yes	ID of the entity. If set, updates the corresponding existing entity.

Responses

200: OK

POST /identity/entity/id/{id}

Operation ID: `entity-update-by-id`

Update, read or delete an entity using entity ID

Parameters

Parameter	Type	Location	Required	Description
`id`	string	path	yes	ID of the entity. If set, updates the corresponding existing entity.

Request body parameters

Parameter	Type	Required	Description
`disabled`	boolean	no	If set true, tokens tied to this identity will not be able to be used (but will not be revoked).
`metadata`	object	no	Metadata to be associated with the entity. In CLI, this parameter can be repeated multiple times, and it all gets merged together. For example: stronghold metadata=key1=value1 metadata=key2=value2
`name`	string	no	Name of the entity
`policies`	array	no	Policies to be tied to the entity.

Responses

200: OK

DELETE /identity/entity/id/{id}

Operation ID: `entity-delete-by-id`

Update, read or delete an entity using entity ID

Parameters

Parameter	Type	Location	Required	Description
`id`	string	path	yes	ID of the entity. If set, updates the corresponding existing entity.

Responses

204: empty body

POST /identity/entity/merge

Operation ID: `entity-merge`

Merge two or more entities together

Request body parameters

Parameter	Type	Required	Description
`conflicting_alias_ids_to_keep`	array	no	Alias IDs to keep in case of conflicting aliases. Ignored if no conflicting aliases found
`force`	boolean	no	Setting this will follow the 'mine' strategy for merging MFA secrets. If there are secrets of the same type both in entities that are merged from and in entity into which all others are getting merged, secrets in the destination will be unaltered. If not set, this API will throw an error containing all the conflicts.
`from_entity_ids`	array	no	Entity IDs which need to get merged
`to_entity_id`	string	no	Entity ID into which all the other entities need to get merged

Responses

200: OK

GET /identity/entity/name

Operation ID: `entity-list-by-name`

List all the entity names

Parameters

Parameter	Type	Location	Required	Description
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /identity/entity/name/{name}

Operation ID: `entity-read-by-name`

Update, read or delete an entity using entity name

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the entity

Responses

200: OK

POST /identity/entity/name/{name}

Operation ID: `entity-update-by-name`

Update, read or delete an entity using entity name

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the entity

Request body parameters

Parameter	Type	Required	Description
`disabled`	boolean	no	If set true, tokens tied to this identity will not be able to be used (but will not be revoked).
`id`	string	no	ID of the entity. If set, updates the corresponding existing entity.
`metadata`	object	no	Metadata to be associated with the entity. In CLI, this parameter can be repeated multiple times, and it all gets merged together. For example: stronghold metadata=key1=value1 metadata=key2=value2
`policies`	array	no	Policies to be tied to the entity.

Responses

200: OK

DELETE /identity/entity/name/{name}

Operation ID: `entity-delete-by-name`

Update, read or delete an entity using entity name

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the entity

Responses

204: empty body

POST /identity/group

Operation ID: `group-create`

Create a new group.

Request body parameters

Parameter	Type	Required	Description
`id`	string	no	ID of the group. If set, updates the corresponding existing group.
`member_entity_ids`	array	no	Entity IDs to be assigned as group members.
`member_group_ids`	array	no	Group IDs to be assigned as group members.
`metadata`	object	no	Metadata to be associated with the group. In CLI, this parameter can be repeated multiple times, and it all gets merged together. For example: stronghold metadata=key1=value1 metadata=key2=value2
`name`	string	no	Name of the group.
`policies`	array	no	Policies to be tied to the group.
`type`	string	no	Type of the group, ‘internal’ or ’external’. Defaults to ‘internal’

Responses

200: OK

POST /identity/group-alias

Operation ID: `group-create-alias`

Creates a new group alias, or updates an existing one.

Request body parameters

Parameter	Type	Required	Description
`canonical_id`	string	no	ID of the group to which this is an alias.
`id`	string	no	ID of the group alias.
`mount_accessor`	string	no	Mount accessor to which this alias belongs to.
`name`	string	no	Alias of the group.

Responses

200: OK

GET /identity/group-alias/id

Operation ID: `group-list-aliases-by-id`

List all the group alias IDs.

Parameters

Parameter	Type	Location	Required	Description
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /identity/group-alias/id/{id}

Operation ID: `group-read-alias-by-id`

Parameters

Parameter	Type	Location	Required	Description
`id`	string	path	yes	ID of the group alias.

Responses

200: OK

POST /identity/group-alias/id/{id}

Operation ID: `group-update-alias-by-id`

Parameters

Parameter	Type	Location	Required	Description
`id`	string	path	yes	ID of the group alias.

Request body parameters

Parameter	Type	Required	Description
`canonical_id`	string	no	ID of the group to which this is an alias.
`mount_accessor`	string	no	Mount accessor to which this alias belongs to.
`name`	string	no	Alias of the group.

Responses

200: OK

DELETE /identity/group-alias/id/{id}

Operation ID: `group-delete-alias-by-id`

Parameters

Parameter	Type	Location	Required	Description
`id`	string	path	yes	ID of the group alias.

Responses

204: empty body

GET /identity/group/id

Operation ID: `group-list-by-id`

List all the group IDs.

Parameters

Parameter	Type	Location	Required	Description
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /identity/group/id/{id}

Operation ID: `group-read-by-id`

Update or delete an existing group using its ID.

Parameters

Parameter	Type	Location	Required	Description
`id`	string	path	yes	ID of the group. If set, updates the corresponding existing group.

Responses

200: OK

POST /identity/group/id/{id}

Operation ID: `group-update-by-id`

Update or delete an existing group using its ID.

Parameters

Parameter	Type	Location	Required	Description
`id`	string	path	yes	ID of the group. If set, updates the corresponding existing group.

Request body parameters

Parameter	Type	Required	Description
`member_entity_ids`	array	no	Entity IDs to be assigned as group members.
`member_group_ids`	array	no	Group IDs to be assigned as group members.
`metadata`	object	no	Metadata to be associated with the group. In CLI, this parameter can be repeated multiple times, and it all gets merged together. For example: stronghold metadata=key1=value1 metadata=key2=value2
`name`	string	no	Name of the group.
`policies`	array	no	Policies to be tied to the group.
`type`	string	no	Type of the group, ‘internal’ or ’external’. Defaults to ‘internal’

Responses

200: OK

DELETE /identity/group/id/{id}

Operation ID: `group-delete-by-id`

Update or delete an existing group using its ID.

Parameters

Parameter	Type	Location	Required	Description
`id`	string	path	yes	ID of the group. If set, updates the corresponding existing group.

Responses

204: empty body

GET /identity/group/name

Operation ID: `group-list-by-name`

Parameters

Parameter	Type	Location	Required	Description
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /identity/group/name/{name}

Operation ID: `group-read-by-name`

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the group.

Responses

200: OK

POST /identity/group/name/{name}

Operation ID: `group-update-by-name`

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the group.

Request body parameters

Parameter	Type	Required	Description
`id`	string	no	ID of the group. If set, updates the corresponding existing group.
`member_entity_ids`	array	no	Entity IDs to be assigned as group members.
`member_group_ids`	array	no	Group IDs to be assigned as group members.
`metadata`	object	no	Metadata to be associated with the group. In CLI, this parameter can be repeated multiple times, and it all gets merged together. For example: stronghold metadata=key1=value1 metadata=key2=value2
`policies`	array	no	Policies to be tied to the group.
`type`	string	no	Type of the group, ‘internal’ or ’external’. Defaults to ‘internal’

Responses

200: OK

DELETE /identity/group/name/{name}

Operation ID: `group-delete-by-name`

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the group.

Responses

204: empty body

POST /identity/lookup/entity

Operation ID: `entity-look-up`

Query entities based on various properties.

Request body parameters

Parameter	Type	Required	Description
`alias_id`	string	no	ID of the alias.
`alias_mount_accessor`	string	no	Accessor of the mount to which the alias belongs to. This should be supplied in conjunction with ‘alias_name’.
`alias_name`	string	no	Name of the alias. This should be supplied in conjunction with ‘alias_mount_accessor’.
`id`	string	no	ID of the entity.
`name`	string	no	Name of the entity.

Responses

200: OK

POST /identity/lookup/group

Operation ID: `group-look-up`

Query groups based on various properties.

Request body parameters

Parameter	Type	Required	Description
`alias_id`	string	no	ID of the alias.
`alias_mount_accessor`	string	no	Accessor of the mount to which the alias belongs to. This should be supplied in conjunction with ‘alias_name’.
`alias_name`	string	no	Name of the alias. This should be supplied in conjunction with ‘alias_mount_accessor’.
`id`	string	no	ID of the group.
`name`	string	no	Name of the group.

Responses

200: OK

GET /identity/mfa/login-enforcement

Operation ID: `mfa-list-login-enforcements`

List login enforcements

Parameters

Parameter	Type	Location	Required	Description
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /identity/mfa/login-enforcement/{name}

Operation ID: `mfa-read-login-enforcement`

Read the current login enforcement

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name for this login enforcement configuration

Responses

200: OK

POST /identity/mfa/login-enforcement/{name}

Operation ID: `mfa-write-login-enforcement`

Create or update a login enforcement

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name for this login enforcement configuration

Request body parameters

Parameter	Type	Required	Description
`auth_method_accessors`	array	no	Array of auth mount accessor IDs
`auth_method_types`	array	no	Array of auth mount types
`identity_entity_ids`	array	no	Array of identity entity IDs
`identity_group_ids`	array	no	Array of identity group IDs
`mfa_method_ids`	array	yes	Array of Method IDs that determine what methods will be enforced

Responses

200: OK

DELETE /identity/mfa/login-enforcement/{name}

Operation ID: `mfa-delete-login-enforcement`

Delete a login enforcement

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name for this login enforcement configuration

Responses

204: empty body

GET /identity/mfa/method

Operation ID: `mfa-list-methods`

List MFA method configurations for all MFA methods

Parameters

Parameter	Type	Location	Required	Description
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /identity/mfa/method/duo

Operation ID: `mfa-list-duo-methods`

List MFA method configurations for the given MFA method

Parameters

Parameter	Type	Location	Required	Description
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

POST /identity/mfa/method/duo

Operation ID: `mfa-create-duo-method`

Create the given MFA method

Request body parameters

Parameter	Type	Required	Description
`api_hostname`	string	no	API host name for Duo.
`integration_key`	string	no	Integration key for Duo.
`method_name`	string	no	The unique name identifier for this MFA method.
`push_info`	string	no	Push information for Duo.
`secret_key`	string	no	Secret key for Duo.
`use_passcode`	boolean	no	If true, the user is reminded to use the passcode upon MFA validation. This option does not enforce using the passcode. Defaults to false.
`username_format`	string	no	A template string for mapping Identity names to MFA method names. Values to subtitute should be placed in {{}}. For example, “{{alias.name}}@example.com”. Currently-supported mappings: alias.name: The name returned by the mount configured via the mount_accessor parameter If blank, the Alias’s name field will be used as-is.

Responses

200: OK

GET /identity/mfa/method/duo/{method_id}

Operation ID: `mfa-read-duo-method-configuration`

Read the current configuration for the given MFA method

Parameters

Parameter	Type	Location	Required	Description
`method_id`	string	path	yes	The unique identifier for this MFA method.

Responses

200: OK

POST /identity/mfa/method/duo/{method_id}

Operation ID: `mfa-configure-duo-method`

Update the configuration for the given MFA method

Parameters

Parameter	Type	Location	Required	Description
`method_id`	string	path	yes	The unique identifier for this MFA method.

Request body parameters

Parameter	Type	Required	Description
`api_hostname`	string	no	API host name for Duo.
`integration_key`	string	no	Integration key for Duo.
`method_name`	string	no	The unique name identifier for this MFA method.
`push_info`	string	no	Push information for Duo.
`secret_key`	string	no	Secret key for Duo.
`use_passcode`	boolean	no	If true, the user is reminded to use the passcode upon MFA validation. This option does not enforce using the passcode. Defaults to false.
`username_format`	string	no	A template string for mapping Identity names to MFA method names. Values to subtitute should be placed in {{}}. For example, “{{alias.name}}@example.com”. Currently-supported mappings: alias.name: The name returned by the mount configured via the mount_accessor parameter If blank, the Alias’s name field will be used as-is.

Responses

200: OK

DELETE /identity/mfa/method/duo/{method_id}

Operation ID: `mfa-delete-duo-method`

Delete a configuration for the given MFA method

Parameters

Parameter	Type	Location	Required	Description
`method_id`	string	path	yes	The unique identifier for this MFA method.

Responses

204: empty body

GET /identity/mfa/method/okta

Operation ID: `mfa-list-okta-methods`

List MFA method configurations for the given MFA method

Parameters

Parameter	Type	Location	Required	Description
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

POST /identity/mfa/method/okta

Operation ID: `mfa-create-okta-method`

Create the given MFA method

Request body parameters

Parameter	Type	Required	Description
`api_token`	string	no	Okta API key.
`base_url`	string	no	The base domain to use for the Okta API. When not specified in the configuration, “okta.com” is used.
`method_name`	string	no	The unique name identifier for this MFA method.
`org_name`	string	no	Name of the organization to be used in the Okta API.
`primary_email`	boolean	no	If true, the username will only match the primary email for the account. Defaults to false.
`production`	boolean	no	(DEPRECATED) Use base_url instead.
`username_format`	string	no	A template string for mapping Identity names to MFA method names. Values to substitute should be placed in {{}}. For example, “{{entity.name}}@example.com”. If blank, the Entity’s name field will be used as-is.

Responses

200: OK

GET /identity/mfa/method/okta/{method_id}

Operation ID: `mfa-read-okta-method-configuration`

Read the current configuration for the given MFA method

Parameters

Parameter	Type	Location	Required	Description
`method_id`	string	path	yes	The unique identifier for this MFA method.

Responses

200: OK

POST /identity/mfa/method/okta/{method_id}

Operation ID: `mfa-configure-okta-method`

Update the configuration for the given MFA method

Parameters

Parameter	Type	Location	Required	Description
`method_id`	string	path	yes	The unique identifier for this MFA method.

Request body parameters

Parameter	Type	Required	Description
`api_token`	string	no	Okta API key.
`base_url`	string	no	The base domain to use for the Okta API. When not specified in the configuration, “okta.com” is used.
`method_name`	string	no	The unique name identifier for this MFA method.
`org_name`	string	no	Name of the organization to be used in the Okta API.
`primary_email`	boolean	no	If true, the username will only match the primary email for the account. Defaults to false.
`production`	boolean	no	(DEPRECATED) Use base_url instead.
`username_format`	string	no	A template string for mapping Identity names to MFA method names. Values to substitute should be placed in {{}}. For example, “{{entity.name}}@example.com”. If blank, the Entity’s name field will be used as-is.

Responses

200: OK

DELETE /identity/mfa/method/okta/{method_id}

Operation ID: `mfa-delete-okta-method`

Delete a configuration for the given MFA method

Parameters

Parameter	Type	Location	Required	Description
`method_id`	string	path	yes	The unique identifier for this MFA method.

Responses

204: empty body

GET /identity/mfa/method/pingid

Operation ID: `mfa-list-ping-id-methods`

List MFA method configurations for the given MFA method

Parameters

Parameter	Type	Location	Required	Description
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

POST /identity/mfa/method/pingid

Operation ID: `mfa-create-ping-id-method`

Create the given MFA method

Request body parameters

Parameter	Type	Required	Description
`method_name`	string	no	The unique name identifier for this MFA method.
`settings_file_base64`	string	no	The settings file provided by Ping, Base64-encoded. This must be a settings file suitable for third-party clients, not the PingID SDK or PingFederate.
`username_format`	string	no	A template string for mapping Identity names to MFA method names. Values to subtitute should be placed in {{}}. For example, “{{alias.name}}@example.com”. Currently-supported mappings: alias.name: The name returned by the mount configured via the mount_accessor parameter If blank, the Alias’s name field will be used as-is.

Responses

200: OK

GET /identity/mfa/method/pingid/{method_id}

Operation ID: `mfa-read-ping-id-method-configuration`

Read the current configuration for the given MFA method

Parameters

Parameter	Type	Location	Required	Description
`method_id`	string	path	yes	The unique identifier for this MFA method.

Responses

200: OK

POST /identity/mfa/method/pingid/{method_id}

Operation ID: `mfa-configure-ping-id-method`

Update the configuration for the given MFA method

Parameters

Parameter	Type	Location	Required	Description
`method_id`	string	path	yes	The unique identifier for this MFA method.

Request body parameters

Parameter	Type	Required	Description
`method_name`	string	no	The unique name identifier for this MFA method.
`settings_file_base64`	string	no	The settings file provided by Ping, Base64-encoded. This must be a settings file suitable for third-party clients, not the PingID SDK or PingFederate.
`username_format`	string	no	A template string for mapping Identity names to MFA method names. Values to subtitute should be placed in {{}}. For example, “{{alias.name}}@example.com”. Currently-supported mappings: alias.name: The name returned by the mount configured via the mount_accessor parameter If blank, the Alias’s name field will be used as-is.

Responses

200: OK

DELETE /identity/mfa/method/pingid/{method_id}

Operation ID: `mfa-delete-ping-id-method`

Delete a configuration for the given MFA method

Parameters

Parameter	Type	Location	Required	Description
`method_id`	string	path	yes	The unique identifier for this MFA method.

Responses

204: empty body

GET /identity/mfa/method/totp

Operation ID: `mfa-list-totp-methods`

List MFA method configurations for the given MFA method

Parameters

Parameter	Type	Location	Required	Description
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

POST /identity/mfa/method/totp

Operation ID: `mfa-create-totp-method`

Create the given MFA method

Request body parameters

Parameter	Type	Required	Description
`algorithm`	string (default: SHA1)	no	The hashing algorithm used to generate the TOTP token. Options include SHA1, SHA256 and SHA512.
`digits`	integer (default: 6)	no	The number of digits in the generated TOTP token. This value can either be 6 or 8.
`issuer`	string	no	The name of the key's issuing organization.
`key_size`	integer (default: 20)	no	Determines the size in bytes of the generated key.
`max_validation_attempts`	integer	no	Max number of allowed validation attempts.
`method_name`	string	no	The unique name identifier for this MFA method.
`period`	integer (default: 30)	no	The length of time used to generate a counter for the TOTP token calculation.
`qr_size`	integer (default: 200)	no	The pixel size of the generated square QR code.
`skew`	integer (default: 1)	no	The number of delay periods that are allowed when validating a TOTP token. This value can either be 0 or 1.

Responses

200: OK

POST /identity/mfa/method/totp/admin-destroy

Operation ID: `mfa-admin-destroy-totp-secret`

Destroys a TOTP secret for the given MFA method ID on the given entity

Request body parameters

Parameter	Type	Required	Description
`entity_id`	string	yes	Identifier of the entity from which the MFA method secret needs to be removed.
`method_id`	string	yes	The unique identifier for this MFA method.

Responses

200: OK

POST /identity/mfa/method/totp/admin-generate

Operation ID: `mfa-admin-generate-totp-secret`

Update or create TOTP secret for the given method ID on the given entity.

Request body parameters

Parameter	Type	Required	Description
`entity_id`	string	yes	Entity ID on which the generated secret needs to get stored.
`method_id`	string	yes	The unique identifier for this MFA method.

Responses

200: OK

POST /identity/mfa/method/totp/generate

Operation ID: `mfa-generate-totp-secret`

Update or create TOTP secret for the given method ID on the given entity.

Request body parameters

Parameter	Type	Required	Description
`method_id`	string	yes	The unique identifier for this MFA method.

Responses

200: OK

GET /identity/mfa/method/totp/{method_id}

Operation ID: `mfa-read-totp-method-configuration`

Read the current configuration for the given MFA method

Parameters

Parameter	Type	Location	Required	Description
`method_id`	string	path	yes	The unique identifier for this MFA method.

Responses

200: OK

POST /identity/mfa/method/totp/{method_id}

Operation ID: `mfa-configure-totp-method`

Update the configuration for the given MFA method

Parameters

Parameter	Type	Location	Required	Description
`method_id`	string	path	yes	The unique identifier for this MFA method.

Request body parameters

Parameter	Type	Required	Description
`algorithm`	string (default: SHA1)	no	The hashing algorithm used to generate the TOTP token. Options include SHA1, SHA256 and SHA512.
`digits`	integer (default: 6)	no	The number of digits in the generated TOTP token. This value can either be 6 or 8.
`issuer`	string	no	The name of the key's issuing organization.
`key_size`	integer (default: 20)	no	Determines the size in bytes of the generated key.
`max_validation_attempts`	integer	no	Max number of allowed validation attempts.
`method_name`	string	no	The unique name identifier for this MFA method.
`period`	integer (default: 30)	no	The length of time used to generate a counter for the TOTP token calculation.
`qr_size`	integer (default: 200)	no	The pixel size of the generated square QR code.
`skew`	integer (default: 1)	no	The number of delay periods that are allowed when validating a TOTP token. This value can either be 0 or 1.

Responses

200: OK

DELETE /identity/mfa/method/totp/{method_id}

Operation ID: `mfa-delete-totp-method`

Delete a configuration for the given MFA method

Parameters

Parameter	Type	Location	Required	Description
`method_id`	string	path	yes	The unique identifier for this MFA method.

Responses

204: empty body

GET /identity/mfa/method/{method_id}

Operation ID: `mfa-read-method-configuration`

Read the current configuration for the given ID regardless of the MFA method type

Parameters

Parameter	Type	Location	Required	Description
`method_id`	string	path	yes	The unique identifier for this MFA method.

Responses

200: OK

GET /identity/oidc/.well-known/keys

Operation ID: `oidc-read-public-keys`

Retrieve public keys

Available without authentication: yes

Responses

200: OK

GET /identity/oidc/.well-known/openid-configuration

Operation ID: `oidc-read-open-id-configuration`

Query OIDC configurations

Available without authentication: yes

Responses

200: OK

GET /identity/oidc/assignment

Operation ID: `oidc-list-assignments`

List OIDC assignments

Parameters

Parameter	Type	Location	Required	Description
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /identity/oidc/assignment/{name}

Operation ID: `oidc-read-assignment`

CRUD operations for OIDC assignments.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the assignment

Responses

200: OK

POST /identity/oidc/assignment/{name}

Operation ID: `oidc-write-assignment`

CRUD operations for OIDC assignments.

Creation supported: yes

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the assignment

Request body parameters

Parameter	Type	Required	Description
`entity_ids`	array	no	Comma separated string or array of identity entity IDs
`group_ids`	array	no	Comma separated string or array of identity group IDs

Responses

200: OK

DELETE /identity/oidc/assignment/{name}

Operation ID: `oidc-delete-assignment`

CRUD operations for OIDC assignments.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the assignment

Responses

204: empty body

GET /identity/oidc/client

Operation ID: `oidc-list-clients`

List OIDC clients

Parameters

Parameter	Type	Location	Required	Description
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /identity/oidc/client/{name}

Operation ID: `oidc-read-client`

CRUD operations for OIDC clients.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the client.

Responses

200: OK

POST /identity/oidc/client/{name}

Operation ID: `oidc-write-client`

CRUD operations for OIDC clients.

Creation supported: yes

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the client.

Request body parameters

Parameter	Type	Required	Description
`access_token_ttl`	integer (default: 24h)	no	The time-to-live for access tokens obtained by the client.
`assignments`	array	no	Comma separated string or array of assignment resources.
`client_type`	string (default: confidential)	no	The client type based on its ability to maintain confidentiality of credentials. The following client types are supported: ‘confidential’, ‘public’. Defaults to ‘confidential’.
`id_token_ttl`	integer (default: 24h)	no	The time-to-live for ID tokens obtained by the client.
`key`	string (default: default)	no	A reference to a named key resource. Cannot be modified after creation. Defaults to the ‘default’ key.
`redirect_uris`	array	no	Comma separated string or array of redirect URIs used by the client. One of these values must exactly match the redirect_uri parameter value used in each authentication request.

Responses

200: OK

DELETE /identity/oidc/client/{name}

Operation ID: `oidc-delete-client`

CRUD operations for OIDC clients.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the client.

Responses

204: empty body

GET /identity/oidc/config

Operation ID: `oidc-read-configuration`

OIDC configuration

Responses

200: OK

POST /identity/oidc/config

Operation ID: `oidc-configure`

OIDC configuration

Request body parameters

Parameter	Type	Required	Description
`issuer`	string	no	Issuer URL to be used in the iss claim of the token. If not set, app_addr will be used.

Responses

200: OK

POST /identity/oidc/introspect

Operation ID: `oidc-introspect`

Verify the authenticity of an OIDC token

Request body parameters

Parameter	Type	Required	Description
`client_id`	string	no	Optional client_id to verify
`token`	string	no	Token to verify

Responses

200: OK

GET /identity/oidc/key

Operation ID: `oidc-list-keys`

List OIDC keys

Parameters

Parameter	Type	Location	Required	Description
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /identity/oidc/key/{name}

Operation ID: `oidc-read-key`

CRUD operations for OIDC keys.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the key

Responses

200: OK

POST /identity/oidc/key/{name}

Operation ID: `oidc-write-key`

CRUD operations for OIDC keys.

Creation supported: yes

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the key

Request body parameters

Parameter	Type	Required	Description
`algorithm`	string (default: RS256)	no	Signing algorithm to use. This will default to RS256.
`allowed_client_ids`	array	no	Comma separated string or array of role client ids allowed to use this key for signing. If empty no roles are allowed. If “*” all roles are allowed.
`rotation_period`	integer (default: 24h)	no	How often to generate a new keypair.
`verification_ttl`	integer (default: 24h)	no	Controls how long the public portion of a key will be available for verification after being rotated.

Responses

200: OK

DELETE /identity/oidc/key/{name}

Operation ID: `oidc-delete-key`

CRUD operations for OIDC keys.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the key

Responses

204: empty body

POST /identity/oidc/key/{name}/rotate

Operation ID: `oidc-rotate-key`

Rotate a named OIDC key.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the key

Request body parameters

Parameter	Type	Required	Description
`verification_ttl`	integer	no	Controls how long the public portion of a key will be available for verification after being rotated. Setting verification_ttl here will override the verification_ttl set on the key.

Responses

200: OK

GET /identity/oidc/provider

Operation ID: `oidc-list-providers`

List OIDC providers

Parameters

Parameter	Type	Location	Required	Description
`allowed_client_id`	string	query	no	Filters the list of OIDC providers to those that allow the given client ID in their set of allowed_client_ids.
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /identity/oidc/provider/{name}

Operation ID: `oidc-read-provider`

CRUD operations for OIDC providers.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the provider

Responses

200: OK

POST /identity/oidc/provider/{name}

Operation ID: `oidc-write-provider`

CRUD operations for OIDC providers.

Creation supported: yes

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the provider

Request body parameters

Parameter	Type	Required	Description
`allowed_client_ids`	array	no	The client IDs that are permitted to use the provider
`issuer`	string	no	Specifies what will be used for the iss claim of ID tokens.
`scopes_supported`	array	no	The scopes supported for requesting on the provider

Responses

200: OK

DELETE /identity/oidc/provider/{name}

Operation ID: `oidc-delete-provider`

CRUD operations for OIDC providers.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the provider

Responses

204: empty body

GET /identity/oidc/provider/{name}/.well-known/keys

Operation ID: `oidc-read-provider-public-keys`

Retrieve public keys

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the provider

Responses

200: OK

GET /identity/oidc/provider/{name}/.well-known/openid-configuration

Operation ID: `oidc-read-provider-open-id-configuration`

Query OIDC configurations

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the provider

Responses

200: OK

GET /identity/oidc/provider/{name}/authorize

Operation ID: `oidc-provider-authorize`

Provides the OIDC Authorization Endpoint.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the provider

Responses

200: OK

POST /identity/oidc/provider/{name}/authorize

Operation ID: `oidc-provider-authorize-with-parameters`

Provides the OIDC Authorization Endpoint.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the provider

Request body parameters

Parameter	Type	Required	Description
`client_id`	string	yes	The ID of the requesting client.
`code_challenge`	string	no	The code challenge derived from the code verifier.
`code_challenge_method`	string (default: plain)	no	The method that was used to derive the code challenge. The following methods are supported: ‘S256’, ‘plain’. Defaults to ‘plain’.
`max_age`	integer	no	The allowable elapsed time in seconds since the last time the end-user was actively authenticated.
`nonce`	string	no	The value that will be returned in the ID token nonce claim after a token exchange.
`redirect_uri`	string	yes	The redirection URI to which the response will be sent.
`response_type`	string	yes	The OIDC authentication flow to be used. The following response types are supported: ‘code’
`scope`	string	yes	A space-delimited, case-sensitive list of scopes to be requested. The ‘openid’ scope is required.
`state`	string	no	The value used to maintain state between the authentication request and client.

Responses

200: OK

POST /identity/oidc/provider/{name}/token

Operation ID: `oidc-provider-token`

Provides the OIDC Token Endpoint.

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the provider

Request body parameters

Parameter	Type	Required	Description
`client_id`	string	no	The ID of the requesting client.
`client_secret`	string	no	The secret of the requesting client.
`code`	string	yes	The authorization code received from the provider’s authorization endpoint.
`code_verifier`	string	no	The code verifier associated with the authorization code.
`grant_type`	string	yes	The authorization grant type. The following grant types are supported: ‘authorization_code’.
`redirect_uri`	string	yes	The callback location where the authentication response was sent.

Responses

200: OK

GET /identity/oidc/provider/{name}/userinfo

Operation ID: `oidc-provider-user-info`

Provides the OIDC UserInfo Endpoint.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the provider

Responses

200: OK

POST /identity/oidc/provider/{name}/userinfo

Operation ID: `oidc-provider-user-info2`

Provides the OIDC UserInfo Endpoint.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the provider

Responses

200: OK

GET /identity/oidc/role

Operation ID: `oidc-list-roles`

List configured OIDC roles

Parameters

Parameter	Type	Location	Required	Description
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /identity/oidc/role/{name}

Operation ID: `oidc-read-role`

CRUD operations on OIDC Roles

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the role

Responses

200: OK

POST /identity/oidc/role/{name}

Operation ID: `oidc-write-role`

CRUD operations on OIDC Roles

Creation supported: yes

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the role

Request body parameters

Parameter	Type	Required	Description
`client_id`	string	no	Optional client_id
`key`	string	yes	The OIDC key to use for generating tokens. The specified key must already exist.
`template`	string	no	The template string to use for generating tokens. This may be in string-ified JSON or base64 format.
`ttl`	integer (default: 24h)	no	TTL of the tokens generated against the role.

Responses

200: OK

DELETE /identity/oidc/role/{name}

Operation ID: `oidc-delete-role`

CRUD operations on OIDC Roles

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the role

Responses

204: empty body

GET /identity/oidc/scope

Operation ID: `oidc-list-scopes`

List OIDC scopes

Parameters

Parameter	Type	Location	Required	Description
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /identity/oidc/scope/{name}

Operation ID: `oidc-read-scope`

CRUD operations for OIDC scopes.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the scope

Responses

200: OK

POST /identity/oidc/scope/{name}

Operation ID: `oidc-write-scope`

CRUD operations for OIDC scopes.

Creation supported: yes

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the scope

Request body parameters

Parameter	Type	Required	Description
`description`	string	no	The description of the scope
`template`	string	no	The template string to use for the scope. This may be in string-ified JSON or base64 format.

Responses

200: OK

DELETE /identity/oidc/scope/{name}

Operation ID: `oidc-delete-scope`

CRUD operations for OIDC scopes.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the scope

Responses

204: empty body

GET /identity/oidc/token/{name}

Operation ID: `oidc-generate-token`

Generate an OIDC token

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the role

Responses

200: OK

POST /identity/persona

Operation ID: `persona-create`

Create a new alias.

Request body parameters

Parameter	Type	Required	Description
`entity_id`	string	no	Entity ID to which this persona belongs to
`id`	string	no	ID of the persona
`metadata`	object	no	Metadata to be associated with the persona. In CLI, this parameter can be repeated multiple times, and it all gets merged together. For example: stronghold metadata=key1=value1 metadata=key2=value2
`mount_accessor`	string	no	Mount accessor to which this persona belongs to
`name`	string	no	Name of the persona

Responses

200: OK

GET /identity/persona/id

Operation ID: `persona-list-by-id`

List all the alias IDs.

Parameters

Parameter	Type	Location	Required	Description
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /identity/persona/id/{id}

Operation ID: `persona-read-by-id`

Update, read or delete an alias ID.

Parameters

Parameter	Type	Location	Required	Description
`id`	string	path	yes	ID of the persona

Responses

200: OK

POST /identity/persona/id/{id}

Operation ID: `persona-update-by-id`

Update, read or delete an alias ID.

Parameters

Parameter	Type	Location	Required	Description
`id`	string	path	yes	ID of the persona

Request body parameters

Parameter	Type	Required	Description
`entity_id`	string	no	Entity ID to which this persona should be tied to
`metadata`	object	no	Metadata to be associated with the persona. In CLI, this parameter can be repeated multiple times, and it all gets merged together. For example: stronghold metadata=key1=value1 metadata=key2=value2
`mount_accessor`	string	no	Mount accessor to which this persona belongs to
`name`	string	no	Name of the persona

Responses

200: OK

DELETE /identity/persona/id/{id}

Operation ID: `persona-delete-by-id`

Update, read or delete an alias ID.

Parameters

Parameter	Type	Location	Required	Description
`id`	string	path	yes	ID of the persona

Responses

204: empty body

system

system

GET /sys/audit

Operation ID: `auditing-list-enabled-devices`

List the enabled audit devices.

Required sudo: yes

Responses

200: OK

POST /sys/audit-hash/{path}

Operation ID: `auditing-calculate-hash`

The hash of the given string via the given audit backend

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	The name of the backend. Cannot be delimited. Example: “mysql”

Request body parameters

Parameter	Type	Required	Description
`input`	string	no	

Responses

200: OK

Parameter	Type	Required	Description
`hash`	string	no	

POST /sys/audit/{path}

Operation ID: `auditing-enable-device`

Enable a new audit device at the supplied path.

Required sudo: yes

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	The name of the backend. Cannot be delimited. Example: “mysql”

Request body parameters

Parameter	Type	Required	Description
`description`	string	no	User-friendly description for this audit backend.
`local`	boolean (default: False)	no	Mark the mount as a local mount, which is not replicated and is unaffected by replication.
`options`	object	no	Configuration options for the audit backend.
`type`	string	no	The type of the backend. Example: “mysql”

Responses

204: OK

DELETE /sys/audit/{path}

Operation ID: `auditing-disable-device`

Disable the audit device at the given path.

Required sudo: yes

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	The name of the backend. Cannot be delimited. Example: “mysql”

Responses

204: OK

GET /sys/auth

Operation ID: `auth-list-enabled-methods`

List the currently enabled credential backends.

Responses

200: OK

GET /sys/auth/{path}

Operation ID: `auth-read-configuration`

Read the configuration of the auth engine at the given path.

Required sudo: yes

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	The path to mount to. Cannot be delimited. Example: “user”

Responses

200: OK

Parameter	Type	Required	Description
`accessor`	string	no	
`config`	object	no	
`deprecation_status`	string	no	
`description`	string	no	
`external_entropy_access`	boolean	no	
`local`	boolean	no	
`options`	object	no	
`plugin_version`	string	no	
`running_plugin_version`	string	no	
`running_sha256`	string	no	
`seal_wrap`	boolean	no	
`type`	string	no	
`uuid`	string	no	

POST /sys/auth/{path}

Operation ID: `auth-enable-method`

Enables a new auth method.

Required sudo: yes

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	The path to mount to. Cannot be delimited. Example: “user”

Request body parameters

Parameter	Type	Required	Description
`config`	object	no	Configuration for this mount, such as plugin_name.
`description`	string	no	User-friendly description for this credential backend.
`external_entropy_access`	boolean (default: False)	no	Whether to give the mount access to Stronghold's external entropy.
`local`	boolean (default: False)	no	Mark the mount as a local mount, which is not replicated and is unaffected by replication.
`options`	object	no	The options to pass into the backend. Should be a json object with string keys and values.
`plugin_name`	string	no	Name of the auth plugin to use based from the name in the plugin catalog.
`plugin_version`	string	no	The semantic version of the plugin to use.
`seal_wrap`	boolean (default: False)	no	Whether to turn on seal wrapping for the mount.
`type`	string	no	The type of the backend. Example: "userpass"

Responses

204: OK

DELETE /sys/auth/{path}

Operation ID: `auth-disable-method`

Disable the auth method at the given auth path

Required sudo: yes

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	The path to mount to. Cannot be delimited. Example: “user”

Responses

204: OK

GET /sys/auth/{path}/tune

Operation ID: `auth-read-tuning-information`

Reads the given auth path’s configuration.

Required sudo: yes

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	Tune the configuration parameters for an auth path.

Responses

200: OK

Parameter	Type	Required	Description
`allowed_managed_keys`	array	no	
`allowed_response_headers`	array	no	
`audit_non_hmac_request_keys`	array	no	
`audit_non_hmac_response_keys`	array	no	
`default_lease_ttl`	integer	no	
`description`	string	no	
`external_entropy_access`	boolean	no	
`force_no_cache`	boolean	no	
`listing_visibility`	string	no	
`max_lease_ttl`	integer	no	
`options`	object	no	
`passthrough_request_headers`	array	no	
`plugin_version`	string	no	
`token_type`	string	no	
`user_lockout_counter_reset_duration`	integer	no	
`user_lockout_disable`	boolean	no	
`user_lockout_duration`	integer	no	
`user_lockout_threshold`	integer	no	

POST /sys/auth/{path}/tune

Operation ID: `auth-tune-configuration-parameters`

Tune configuration parameters for a given auth path.

Required sudo: yes

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	Tune the configuration parameters for an auth path.

Request body parameters

Parameter	Type	Required	Description
`allowed_response_headers`	array	no	A list of headers to whitelist and allow a plugin to set on responses.
`audit_non_hmac_request_keys`	array	no	The list of keys in the request data object that will not be HMAC’ed by audit devices.
`audit_non_hmac_response_keys`	array	no	The list of keys in the response data object that will not be HMAC’ed by audit devices.
`default_lease_ttl`	string	no	The default lease TTL for this mount.
`description`	string	no	User-friendly description for this credential backend.
`listing_visibility`	string	no	Determines the visibility of the mount in the UI-specific listing endpoint. Accepted value are ‘unauth’ and ‘hidden’, with the empty default (’’) behaving like ‘hidden’.
`max_lease_ttl`	string	no	The max lease TTL for this mount.
`options`	object	no	The options to pass into the backend. Should be a json object with string keys and values.
`passthrough_request_headers`	array	no	A list of headers to whitelist and pass from the request to the plugin.
`plugin_version`	string	no	The semantic version of the plugin to use.
`token_type`	string	no	The type of token to issue (service or batch).
`user_lockout_config`	object	no	The user lockout configuration to pass into the backend. Should be a json object with string keys and values.

Responses

204: OK

POST /sys/capabilities

Operation ID: `query-token-capabilities`

Fetches the capabilities of the given token on the given path.

Request body parameters

Parameter	Type	Required	Description
`path`	array	no	⚠ Deprecated. Use ‘paths’ instead.
`paths`	array	no	Paths on which capabilities are being queried.
`token`	string	no	Token for which capabilities are being queried.

Responses

200: OK

POST /sys/capabilities-accessor

Operation ID: `query-token-accessor-capabilities`

Fetches the capabilities of the token associated with the given token, on the given path.

Request body parameters

Parameter	Type	Required	Description
`accessor`	string	no	Accessor of the token for which capabilities are being queried.
`path`	array	no	⚠ Deprecated. Use ‘paths’ instead.
`paths`	array	no	Paths on which capabilities are being queried.

Responses

200: OK

POST /sys/capabilities-self

Operation ID: `query-token-self-capabilities`

Fetches the capabilities of the given token on the given path.

Request body parameters

Parameter	Type	Required	Description
`path`	array	no	⚠ Deprecated. Use ‘paths’ instead.
`paths`	array	no	Paths on which capabilities are being queried.
`token`	string	no	Token for which capabilities are being queried.

Responses

200: OK

GET /sys/config/auditing/request-headers

Operation ID: `auditing-list-request-headers`

List the request headers that are configured to be audited.

Required sudo: yes

Responses

200: OK

Parameter	Type	Required	Description
`headers`	object	no	

GET /sys/config/auditing/request-headers/{header}

Operation ID: `auditing-read-request-header-information`

List the information for the given request header.

Required sudo: yes

Parameters

Parameter	Type	Location	Required	Description
`header`	string	path	yes	

Responses

200: OK

POST /sys/config/auditing/request-headers/{header}

Operation ID: `auditing-enable-request-header`

Enable auditing of a header.

Required sudo: yes

Parameters

Parameter	Type	Location	Required	Description
`header`	string	path	yes	

Request body parameters

Parameter	Type	Required	Description
`hmac`	boolean	no	

Responses

204: OK

DELETE /sys/config/auditing/request-headers/{header}

Operation ID: `auditing-disable-request-header`

Disable auditing of the given request header.

Required sudo: yes

Parameters

Parameter	Type	Location	Required	Description
`header`	string	path	yes	

Responses

204: OK

GET /sys/config/control-group

Operation ID: `enterprise-stub-read-config-control-group`

Responses

200: OK

POST /sys/config/control-group

Operation ID: `enterprise-stub-write-config-control-group`

Responses

200: OK

DELETE /sys/config/control-group

Operation ID: `enterprise-stub-delete-config-control-group`

Responses

204: empty body

GET /sys/config/cors

Operation ID: `cors-read-configuration`

Return the current CORS settings.

Required sudo: yes

Responses

200: OK

Parameter	Type	Required	Description
`allowed_headers`	array	no	
`allowed_origins`	array	no	
`enabled`	boolean	no	

POST /sys/config/cors

Operation ID: `cors-configure`

Configure the CORS settings.

Required sudo: yes

Request body parameters

Parameter	Type	Required	Description
`allowed_headers`	array	no	A comma-separated string or array of strings indicating headers that are allowed on cross-origin requests.
`allowed_origins`	array	no	A comma-separated string or array of strings indicating origins that may make cross-origin requests.
`enable`	boolean	no	Enables or disables CORS headers on requests.

Responses

204: OK

DELETE /sys/config/cors

Operation ID: `cors-delete-configuration`

Remove any CORS settings.

Required sudo: yes

Responses

204: OK

GET /sys/config/group-policy-application

Operation ID: `enterprise-stub-read-config-group-policy-application`

Responses

200: OK

POST /sys/config/group-policy-application

Operation ID: `enterprise-stub-write-config-group-policy-application`

Responses

200: OK

POST /sys/config/reload/{subsystem}

Operation ID: `reload-subsystem`

Reload the given subsystem

Parameters

Parameter	Type	Location	Required	Description
`subsystem`	string	path	yes	

Responses

204: OK

GET /sys/config/state/sanitized

Operation ID: `read-sanitized-configuration-state`

Return a sanitized version of the Stronghold server configuration.

Responses

200: OK

GET /sys/config/ui/headers

Operation ID: `ui-headers-list`

Return a list of configured UI headers.

Parameters

Parameter	Type	Location	Required	Description
`list`	string (true)	query	yes	Must be set to `true`

Responses

200:

Parameter	Type	Required	Description
`keys`	array	no	Lists of configured UI headers. Omitted if list is empty

GET /sys/config/ui/headers/{header}

Operation ID: `ui-headers-read-configuration`

Return the given UI header's configuration

Required sudo: yes

Parameters

Parameter	Type	Location	Required	Description
`header`	string	path	yes	The name of the header.

Responses

200: OK

Parameter	Type	Required	Description
`value`	string	no	returns the first header value when `multivalue` request parameter is false
`values`	array	no	returns all header values when `multivalue` request parameter is true

POST /sys/config/ui/headers/{header}

Operation ID: `ui-headers-configure`

Configure the values to be returned for the UI header.

Required sudo: yes

Parameters

Parameter	Type	Location	Required	Description
`header`	string	path	yes	The name of the header.

Request body parameters

Parameter	Type	Required	Description
`multivalue`	boolean	no	Returns multiple values if true
`values`	array	no	The values to set the header.

Responses

200: OK

DELETE /sys/config/ui/headers/{header}

Operation ID: `ui-headers-delete-configuration`

Remove a UI header.

Required sudo: yes

Parameters

Parameter	Type	Location	Required	Description
`header`	string	path	yes	The name of the header.

Responses

204: OK

POST /sys/control-group/authorize

Operation ID: `enterprise-stub-write-control-group-authorize`

Responses

200: OK

POST /sys/control-group/request

Operation ID: `enterprise-stub-write-control-group-request`

Responses

200: OK

POST /sys/decode-token

Operation ID: `decode`

Decodes the encoded token with the otp.

Available without authentication: yes

Request body parameters

Parameter	Type	Required	Description
`encoded_token`	string	no	Specifies the encoded token (result from generate-root).
`otp`	string	no	Specifies the otp code for decode.

Responses

200: OK

GET /sys/experiments

Operation ID: `list-experimental-features`

Returns the available and enabled experiments

Responses

200: OK

GET /sys/generate-root

Operation ID: `root-token-generation-read-progress2`

Read the configuration and progress of the current root generation attempt.

Responses

200: OK

Parameter	Type	Required	Description
`complete`	boolean	no	
`encoded_root_token`	string	no	
`encoded_token`	string	no	
`nonce`	string	no	
`otp`	string	no	
`otp_length`	integer	no	
`pgp_fingerprint`	string	no	
`progress`	integer	no	
`required`	integer	no	
`started`	boolean	no	

POST /sys/generate-root

Operation ID: `root-token-generation-initialize-2`

Initializes a new root generation attempt.

Request body parameters

Parameter	Type	Required	Description
`pgp_key`	string	no	Specifies a base64-encoded PGP public key.

Responses

200: OK

Parameter	Type	Required	Description
`complete`	boolean	no	
`encoded_root_token`	string	no	
`encoded_token`	string	no	
`nonce`	string	no	
`otp`	string	no	
`otp_length`	integer	no	
`pgp_fingerprint`	string	no	
`progress`	integer	no	
`required`	integer	no	
`started`	boolean	no	

DELETE /sys/generate-root

Operation ID: `root-token-generation-cancel-2`

Cancels any in-progress root generation attempt.

Responses

204: OK

GET /sys/generate-root/attempt

Operation ID: `root-token-generation-read-progress`

Read the configuration and progress of the current root generation attempt.

Available without authentication: yes

Responses

200: OK

Parameter	Type	Required	Description
`complete`	boolean	no	
`encoded_root_token`	string	no	
`encoded_token`	string	no	
`nonce`	string	no	
`otp`	string	no	
`otp_length`	integer	no	
`pgp_fingerprint`	string	no	
`progress`	integer	no	
`required`	integer	no	
`started`	boolean	no	

POST /sys/generate-root/attempt

Operation ID: `root-token-generation-initialize`

Initializes a new root generation attempt.

Available without authentication: yes

Request body parameters

Parameter	Type	Required	Description
`pgp_key`	string	no	Specifies a base64-encoded PGP public key.

Responses

200: OK

Parameter	Type	Required	Description
`complete`	boolean	no	
`encoded_root_token`	string	no	
`encoded_token`	string	no	
`nonce`	string	no	
`otp`	string	no	
`otp_length`	integer	no	
`pgp_fingerprint`	string	no	
`progress`	integer	no	
`required`	integer	no	
`started`	boolean	no	

DELETE /sys/generate-root/attempt

Operation ID: `root-token-generation-cancel`

Cancels any in-progress root generation attempt.

Available without authentication: yes

Responses

204: OK

POST /sys/generate-root/update

Operation ID: `root-token-generation-update`

Enter a single unseal key share to progress the root generation attempt.

Available without authentication: yes

Request body parameters

Parameter	Type	Required	Description
`key`	string	no	Specifies a single unseal key share.
`nonce`	string	no	Specifies the nonce of the attempt.

Responses

200: OK

Parameter	Type	Required	Description
`complete`	boolean	no	
`encoded_root_token`	string	no	
`encoded_token`	string	no	
`nonce`	string	no	
`otp`	string	no	
`otp_length`	integer	no	
`pgp_fingerprint`	string	no	
`progress`	integer	no	
`required`	integer	no	
`started`	boolean	no	

GET /sys/ha-status

Operation ID: `ha-status`

Check the HA status of a Stronghold cluster

Responses

200: OK

Parameter	Type	Required	Description
`nodes`	array	no	

GET /sys/health

Operation ID: `read-health-status`

Returns the health status of Stronghold.

Available without authentication: yes

Responses

200: initialized, unsealed, and active

429: unsealed and standby

472: data recovery mode replication secondary and active

501: not initialized

503: sealed

GET /sys/host-info

Operation ID: `collect-host-information`

Information about the host instance that this Stronghold server is running on.

Responses

200: OK

Parameter	Type	Required	Description
`cpu`	array	no	
`cpu_times`	array	no	
`disk`	array	no	
`host`	object	no	
`memory`	object	no	
`timestamp`	string	no	

GET /sys/in-flight-req

Operation ID: `collect-in-flight-request-information`

reports in-flight requests

Responses

200: OK

GET /sys/init

Operation ID: `read-initialization-status`

Returns the initialization status of Stronghold.

Available without authentication: yes

Responses

200: OK

POST /sys/init

Operation ID: `initialize`

Initialize a new Stronghold.

Available without authentication: yes

Request body parameters

Parameter	Type	Required	Description
`pgp_keys`	array	no	Specifies an array of PGP public keys used to encrypt the output unseal keys. Ordering is preserved. The keys must be base64-encoded from their original binary representation. The size of this array must be the same as `secret_shares`.
`recovery_pgp_keys`	array	no	Specifies an array of PGP public keys used to encrypt the output recovery keys. Ordering is preserved. The keys must be base64-encoded from their original binary representation. The size of this array must be the same as `recovery_shares`.
`recovery_shares`	integer	no	Specifies the number of shares to split the recovery key into.
`recovery_threshold`	integer	no	Specifies the number of shares required to reconstruct the recovery key. This must be less than or equal to `recovery_shares`.
`root_token_pgp_key`	string	no	Specifies a PGP public key used to encrypt the initial root token. The key must be base64-encoded from its original binary representation.
`secret_shares`	integer	no	Specifies the number of shares to split the unseal key into.
`secret_threshold`	integer	no	Specifies the number of shares required to reconstruct the unseal key. This must be less than or equal secret_shares. If using Stronghold HSM with auto-unsealing, this value must be the same as `secret_shares`.
`stored_shares`	integer	no	Specifies the number of shares that should be encrypted by the HSM and stored for auto-unsealing. Currently must be the same as `secret_shares`.

Responses

200: OK

GET /sys/internal/counters/activity

Operation ID: `internal-client-activity-report-counts`

Report the client count metrics, for this namespace and all child namespaces.

Responses

200: OK

GET /sys/internal/counters/activity/export

Operation ID: `internal-client-activity-export`

Report the client count metrics, for this namespace and all child namespaces.

Responses

200: OK

GET /sys/internal/counters/activity/monthly

Operation ID: `internal-client-activity-report-counts-this-month`

Report the number of clients for this month, for this namespace and all child namespaces.

Responses

200: OK

GET /sys/internal/counters/config

Operation ID: `internal-client-activity-read-configuration`

Read the client count tracking configuration.

Responses

200: OK

POST /sys/internal/counters/config

Operation ID: `internal-client-activity-configure`

Enable or disable collection of client count, set retention period, or set default reporting period.

Request body parameters

Parameter	Type	Required	Description
`default_report_months`	integer (default: 12)	no	Number of months to report if no start date specified.
`enabled`	string (default: default)	no	Enable or disable collection of client count: enable, disable, or default.
`retention_months`	integer (default: 24)	no	Number of months of client data to retain. Setting to 0 will clear all existing data.

Responses

200: OK

GET /sys/internal/counters/entities

Operation ID: `internal-count-entities`

Backwards compatibility is not guaranteed for this API

Responses

200: OK

Parameter	Type	Required	Description
`counters`	object	no	

GET /sys/internal/counters/requests

Operation ID: `internal-count-requests`

Backwards compatibility is not guaranteed for this API

Responses

200: OK

GET /sys/internal/counters/tokens

Operation ID: `internal-count-tokens`

Backwards compatibility is not guaranteed for this API

Responses

200: OK

Parameter	Type	Required	Description
`counters`	object	no	

GET /sys/internal/inspect/router/{tag}

Operation ID: `internal-inspect-router`

Expose the route entry and mount entry tables present in the router

Required sudo: yes

Parameters

Parameter	Type	Location	Required	Description
`tag`	string	path	yes	Name of subtree being observed

Responses

200: OK

GET /sys/internal/specs/openapi

Operation ID: `internal-generate-open-api-document`

Generate an OpenAPI 3 document of all mounted paths.

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`generic_mount_paths`	boolean	query	no	Use generic mount paths

Responses

200: OK

POST /sys/internal/specs/openapi

Operation ID: `internal-generate-open-api-document-with-parameters`

Generate an OpenAPI 3 document of all mounted paths.

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`generic_mount_paths`	boolean	query	no	Use generic mount paths

Request body parameters

Parameter	Type	Required	Description
`context`	string	no	Context string appended to every operationId

Responses

200: OK

GET /sys/internal/ui/feature-flags

Operation ID: `internal-ui-list-enabled-feature-flags`

Lists enabled feature flags.

Responses

200: OK

Parameter	Type	Required	Description
`feature_flags`	array	no	

GET /sys/internal/ui/mounts

Operation ID: `internal-ui-list-enabled-visible-mounts`

Lists all enabled and visible auth and secrets mounts.

Available without authentication: yes

Responses

200: OK

Parameter	Type	Required	Description
`auth`	object	no	auth mounts
`secret`	object	no	secret mounts

GET /sys/internal/ui/mounts/{path}

Operation ID: `internal-ui-read-mount-information`

Return information about the given mount.

Available without authentication: yes

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	The path of the mount.

Responses

200: OK

Parameter	Type	Required	Description
`accessor`	string	no	
`config`	object	no	
`description`	string	no	
`external_entropy_access`	boolean	no	
`local`	boolean	no	
`options`	object	no	
`path`	string	no	
`plugin_version`	string	no	
`running_plugin_version`	string	no	
`running_sha256`	string	no	
`seal_wrap`	boolean	no	
`type`	string	no	
`uuid`	string	no	

GET /sys/internal/ui/namespaces

Operation ID: `internal-ui-list-namespaces`

Backwards compatibility is not guaranteed for this API

Available without authentication: yes

Responses

200: OK

Parameter	Type	Required	Description
`keys`	array	no	field is only returned if there are one or more namespaces

GET /sys/internal/ui/resultant-acl

Operation ID: `internal-ui-read-resultant-acl`

Backwards compatibility is not guaranteed for this API

Responses

200: OK

Parameter	Type	Required	Description
`exact_paths`	object	no	
`glob_paths`	object	no	
`root`	boolean	no	

204: empty response returned if no client token

GET /sys/key-status

Operation ID: `encryption-key-status`

Provides information about the backend encryption key.

Responses

200: OK

GET /sys/leader

Operation ID: `leader-status`

Returns the high availability status and current leader instance of Stronghold.

Available without authentication: yes

Responses

200: OK

Parameter	Type	Required	Description
`active_time`	string	no	
`ha_enabled`	boolean	no	
`is_self`	boolean	no	
`last_wal`	integer	no	
`leader_address`	string	no	
`leader_cluster_address`	string	no	
`performance_standby`	boolean	no	
`performance_standby_last_remote_wal`	integer	no	
`raft_applied_index`	integer	no	
`raft_committed_index`	integer	no	

GET /sys/leases

Operation ID: `leases-list`

List leases associated with this Stronghold cluster

Required sudo: yes

Responses

200: OK

Parameter	Type	Required	Description
`counts`	integer	no	Number of matching leases per mount
`lease_count`	integer	no	Number of matching leases

GET /sys/leases/count

Operation ID: `leases-count`

Count of leases associated with this Stronghold cluster

Responses

200: OK

Parameter	Type	Required	Description
`counts`	integer	no	Number of matching leases per mount
`lease_count`	integer	no	Number of matching leases

POST /sys/leases/lookup

Operation ID: `leases-read-lease`

View or list lease metadata.

Request body parameters

Parameter	Type	Required	Description
`lease_id`	string	no	The lease identifier to renew. This is included with a lease.

Responses

200: OK

Parameter	Type	Required	Description
`expire_time`	string	no	Optional lease expiry time
`id`	string	no	Lease id
`issue_time`	string	no	Timestamp for the lease’s issue time
`last_renewal`	string	no	Optional Timestamp of the last time the lease was renewed
`renewable`	boolean	no	True if the lease is able to be renewed
`ttl`	integer	no	Time to Live set for the lease, returns 0 if unset

GET /sys/leases/lookup/

Operation ID: `leases-look-up`

View or list lease metadata.

Required sudo: yes

Parameters

Parameter	Type	Location	Required	Description
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

Parameter	Type	Required	Description
`keys`	array	no	A list of lease ids

GET /sys/leases/lookup/{prefix}

Operation ID: `leases-look-up-with-prefix`

View or list lease metadata.

Required sudo: yes

Parameters

Parameter	Type	Location	Required	Description
`prefix`	string	path	yes	The path to list leases under. Example: “aws/creds/deploy”
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

Parameter	Type	Required	Description
`keys`	array	no	A list of lease ids

POST /sys/leases/renew

Operation ID: `leases-renew-lease`

Renews a lease, requesting to extend the lease.

Request body parameters

Parameter	Type	Required	Description
`increment`	integer	no	The desired increment in seconds to the lease
`lease_id`	string	no	The lease identifier to renew. This is included with a lease.
`url_lease_id`	string	no	The lease identifier to renew. This is included with a lease.

Responses

204: OK

POST /sys/leases/renew/{url_lease_id}

Operation ID: `leases-renew-lease-with-id`

Renews a lease, requesting to extend the lease.

Parameters

Parameter	Type	Location	Required	Description
`url_lease_id`	string	path	yes	The lease identifier to renew. This is included with a lease.

Request body parameters

Parameter	Type	Required	Description
`increment`	integer	no	The desired increment in seconds to the lease
`lease_id`	string	no	The lease identifier to renew. This is included with a lease.

Responses

204: OK

POST /sys/leases/revoke

Operation ID: `leases-revoke-lease`

Revokes a lease immediately.

Request body parameters

Parameter	Type	Required	Description
`lease_id`	string	no	The lease identifier to renew. This is included with a lease.
`sync`	boolean (default: True)	no	Whether or not to perform the revocation synchronously
`url_lease_id`	string	no	The lease identifier to renew. This is included with a lease.

Responses

204: OK

POST /sys/leases/revoke-force/{prefix}

Operation ID: `leases-force-revoke-lease-with-prefix`

Revokes all secrets or tokens generated under a given prefix immediately

Required sudo: yes

Parameters

Parameter	Type	Location	Required	Description
`prefix`	string	path	yes	The path to revoke keys under. Example: “prod/aws/ops”

Responses

204: OK

POST /sys/leases/revoke-prefix/{prefix}

Operation ID: `leases-revoke-lease-with-prefix`

Revokes all secrets (via a lease ID prefix) or tokens (via the tokens’ path property) generated under a given prefix immediately.

Required sudo: yes

Parameters

Parameter	Type	Location	Required	Description
`prefix`	string	path	yes	The path to revoke keys under. Example: “prod/aws/ops”

Request body parameters

Parameter	Type	Required	Description
`sync`	boolean (default: True)	no	Whether or not to perform the revocation synchronously

Responses

204: OK

POST /sys/leases/revoke/{url_lease_id}

Operation ID: `leases-revoke-lease-with-id`

Revokes a lease immediately.

Parameters

Parameter	Type	Location	Required	Description
`url_lease_id`	string	path	yes	The lease identifier to renew. This is included with a lease.

Request body parameters

Parameter	Type	Required	Description
`lease_id`	string	no	The lease identifier to renew. This is included with a lease.
`sync`	boolean (default: True)	no	Whether or not to perform the revocation synchronously

Responses

204: OK

POST /sys/leases/tidy

Operation ID: `leases-tidy`

This endpoint performs cleanup tasks that can be run if certain error conditions have occurred.

Responses

204: OK

GET /sys/locked-users

Operation ID: `locked-users-list`

Report the locked user count metrics, for this namespace and all child namespaces.

Responses

200: OK

POST /sys/locked-users/{mount_accessor}/unlock/{alias_identifier}

Operation ID: `locked-users-unlock`

Unlocks the user with given mount_accessor and alias_identifier

Parameters

Parameter	Type	Location	Required	Description
`alias_identifier`	string	path	yes	It is the name of the alias (user). For example, if the alias belongs to userpass backend, the name should be a valid username within userpass auth method. If the alias belongs to an approle auth method, the name should be a valid RoleID
`mount_accessor`	string	path	yes	MountAccessor is the identifier of the mount entry to which the user belongs

Responses

200: OK

GET /sys/loggers

Operation ID: `loggers-read-verbosity-level`

Read the log level for all existing loggers.

Responses

200: OK

POST /sys/loggers

Operation ID: `loggers-update-verbosity-level`

Modify the log level for all existing loggers.

Request body parameters

Parameter	Type	Required	Description
`level`	string	no	Log verbosity level. Supported values (in order of detail) are “trace”, “debug”, “info”, “warn”, and “error”.

Responses

204: OK

DELETE /sys/loggers

Operation ID: `loggers-revert-verbosity-level`

Revert the all loggers to use log level provided in config.

Responses

204: OK

GET /sys/loggers/{name}

Operation ID: `loggers-read-verbosity-level-for`

Read the log level for a single logger.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	The name of the logger to be modified.

Responses

200: OK

POST /sys/loggers/{name}

Operation ID: `loggers-update-verbosity-level-for`

Modify the log level of a single logger.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	The name of the logger to be modified.

Request body parameters

Parameter	Type	Required	Description
`level`	string	no	Log verbosity level. Supported values (in order of detail) are “trace”, “debug”, “info”, “warn”, and “error”.

Responses

204: OK

DELETE /sys/loggers/{name}

Operation ID: `loggers-revert-verbosity-level-for`

Revert a single logger to use log level provided in config.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	The name of the logger to be modified.

Responses

204: OK

GET /sys/managed-keys/{type}

Operation ID: `managed-keys-list-keys`

List managed keys of a given type

Parameters

Parameter	Type	Location	Required	Description
`type`	string	path	yes	The type of managed key (pkcs11, yandexcloudkms)
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

Parameter	Type	Required	Description
`keys`	array	no	

GET /sys/managed-keys/{type}/{name}

Operation ID: `managed-keys-read-key`

Read managed key configuration

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	The name of the managed key
`type`	string	path	yes	The type of managed key (pkcs11, yandexcloudkms)

Responses

200: OK

POST /sys/managed-keys/{type}/{name}

Operation ID: `managed-keys-update`

Update a managed key

Creation supported: yes

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	The name of the managed key
`type`	string	path	yes	The type of managed key (pkcs11, yandexcloudkms)

Request body parameters

Parameter	Type	Required	Description
`allow_generate_key`	boolean	no	If true, allows users of the key to trigger key generation.
`allow_replace_key`	boolean	no	If true, allows users of the key to provide key material which may replace keys that were previously present.
`allow_store_key`	boolean	no	If true, allows users of the key to provide key material where none was present.
`any_mount`	boolean	no	If true, this key may be accessed by any mount without the mount's allowed_manage_keys field being set.
`endpoint`	string	no	[yandexcloudkms] Custom Yandex Cloud API endpoint (optional).
`force_rw_session`	boolean (default: False)	no	[pkcs11] If true, forces read/write sessions on the HSM.
`key_bits`	integer	no	[pkcs11] For RSA keys, the desired key length in bits (2048, 3072, or 4096).
`key_id`	string	no	[pkcs11] The id of a PKCS#11 key to use.
`key_label`	string	no	[pkcs11] The label of a PKCS#11 key to use.
`kms_key_id`	string	no	[yandexcloudkms] The ID of the symmetric key in Yandex Cloud KMS.
`library`	string	no	[pkcs11] The name of a kms_library stanza from server configuration.
`max_parallel`	integer	no	[pkcs11] The maximum number of concurrent operations to the HSM.
`mechanism`	string	no	[pkcs11] The mechanism for the given key, specified as a decimal or hexadecimal (prefixed by 0x) string.
`oauth_token`	string	no	[yandexcloudkms] OAuth token for Yandex Cloud authentication. Mutually exclusive with service_account_key_json.

Parameter	Type	Required	Description
`pin`	string	no	[pkcs11] The access PIN for the slot.
`reverse_signature`	boolean (default: False)	no	[pkcs11] If true, reverses GOST signature bytes from the HSM (CryptoPro compatibility).
`service_account_key_json`	string	no	[yandexcloudkms] JSON of the authorized key for a Yandex Cloud service account. Stored inside Vault, not on the filesystem. Mutually exclusive with oauth_token.
`slot`	string	no	[pkcs11] The slot number to use, specified as a string (e.g. “0”).
`token_label`	string	no	[pkcs11] The slot token label to use.
`usages`	string	no	A comma-delimited list of the allowed usages of this key. Valid values are encrypt, decrypt, sign, verify, wrap, unwrap, mac, and random.

Responses

204: OK

DELETE /sys/managed-keys/{type}/{name}

Operation ID: `managed-keys-delete-key`

Delete a managed key

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	The name of the managed key
`type`	string	path	yes	The type of managed key (pkcs11, yandexcloudkms)

Responses

204: OK

POST /sys/managed-keys/{type}/{name}/test/sign

Operation ID: `managed-keys-test-sign`

Test managed key signing

Creation supported: yes

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	The name of the managed key
`type`	string	path	yes	The type of managed key (pkcs11, yandexcloudkms)

Request body parameters

Parameter	Type	Required	Description
`hash_algorithm`	string (default:)	no	The hashing algorithm to use when signing/ verifying the random data.
`use_pss`	boolean (default: False)	no	For RSA backed managed keys attempt to sign with PSS

Responses

204: OK

GET /sys/metrics

Operation ID: `metrics`

Export the metrics aggregated for telemetry purpose.

Parameters

Parameter	Type	Location	Required	Description
`format`	string	query	no	Format to export metrics into. Currently accepts only “prometheus”.

Responses

200: OK

GET /sys/mfa/method

Operation ID: `enterprise-stub-list-mfa-method`

Parameters

Parameter	Type	Location	Required	Description
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /sys/mfa/method/duo/{name}

Operation ID: `enterprise-stub-read-mfa-method-duo-name`

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	

Responses

200: OK

POST /sys/mfa/method/duo/{name}

Operation ID: `enterprise-stub-write-mfa-method-duo-name`

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	

Responses

200: OK

DELETE /sys/mfa/method/duo/{name}

Operation ID: `enterprise-stub-delete-mfa-method-duo-name`

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	

Responses

204: empty body

GET /sys/mfa/method/okta/{name}

Operation ID: `enterprise-stub-read-mfa-method-okta-name`

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	

Responses

200: OK

POST /sys/mfa/method/okta/{name}

Operation ID: `enterprise-stub-write-mfa-method-okta-name`

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	

Responses

200: OK

DELETE /sys/mfa/method/okta/{name}

Operation ID: `enterprise-stub-delete-mfa-method-okta-name`

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	

Responses

204: empty body

GET /sys/mfa/method/pingid/{name}

Operation ID: `enterprise-stub-read-mfa-method-pingid-name`

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	

Responses

200: OK

POST /sys/mfa/method/pingid/{name}

Operation ID: `enterprise-stub-write-mfa-method-pingid-name`

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	

Responses

200: OK

DELETE /sys/mfa/method/pingid/{name}

Operation ID: `enterprise-stub-delete-mfa-method-pingid-name`

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	

Responses

204: empty body

GET /sys/mfa/method/totp/{name}

Operation ID: `enterprise-stub-read-mfa-method-totp-name`

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	

Responses

200: OK

POST /sys/mfa/method/totp/{name}

Operation ID: `enterprise-stub-write-mfa-method-totp-name`

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	

Responses

200: OK

DELETE /sys/mfa/method/totp/{name}

Operation ID: `enterprise-stub-delete-mfa-method-totp-name`

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	

Responses

204: empty body

POST /sys/mfa/method/totp/{name}/admin-destroy

Operation ID: `enterprise-stub-write-mfa-method-totp-name-admin-destroy`

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	

Responses

200: OK

POST /sys/mfa/method/totp/{name}/admin-generate

Operation ID: `enterprise-stub-write-mfa-method-totp-name-admin-generate`

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	

Responses

200: OK

GET /sys/mfa/method/totp/{name}/generate

Operation ID: `enterprise-stub-read-mfa-method-totp-name-generate`

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	

Responses

200: OK

POST /sys/mfa/validate

Operation ID: `mfa-validate`

Validates the login for the given MFA methods. Upon successful validation, it returns an auth response containing the client token

Available without authentication: yes

Request body parameters

Parameter	Type	Required	Description
`mfa_payload`	object	yes	A map from MFA method ID to a slice of passcodes or an empty slice if the method does not use passcodes
`mfa_request_id`	string	yes	ID for this MFA request

Responses

200: OK

GET /sys/monitor

Operation ID: `monitor`

Parameters

Parameter	Type	Location	Required	Description
`log_format`	string	query	no	Output format of logs. Supported values are “standard” and “json”. The default is “standard”.
`log_level`	string	query	no	Log level to view system logs at. Currently supported values are “trace”, “debug”, “info”, “warn”, “error”.

Responses

200: OK

GET /sys/mounts

Operation ID: `mounts-list-secrets-engines`

List the currently mounted backends.

Responses

200: OK

GET /sys/mounts/{path}

Operation ID: `mounts-read-configuration`

Read the configuration of the secret engine at the given path.

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	The path to mount to. Example: “aws/east”

Responses

200: OK

Parameter	Type	Required	Description
`accessor`	string	no	
`config`	object	no	Configuration for this mount, such as default_lease_ttl and max_lease_ttl.
`deprecation_status`	string	no	
`description`	string	no	User-friendly description for this mount.
`external_entropy_access`	boolean	no	
`local`	boolean (default: False)	no	Mark the mount as a local mount, which is not replicated and is unaffected by replication.
`options`	object	no	The options to pass into the backend. Should be a json object with string keys and values.
`plugin_version`	string	no	The semantic version of the plugin to use.
`running_plugin_version`	string	no	
`running_sha256`	string	no	
`seal_wrap`	boolean (default: False)	no	Whether to turn on seal wrapping for the mount.
`type`	string	no	The type of the backend. Example: “passthrough”
`uuid`	string	no	

POST /sys/mounts/{path}

Operation ID: `mounts-enable-secrets-engine`

Enable a new secrets engine at the given path.

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	The path to mount to. Example: “aws/east”

Request body parameters

Parameter	Type	Required	Description
`config`	object	no	Configuration for this mount, such as default_lease_ttl and max_lease_ttl.
`description`	string	no	User-friendly description for this mount.
`external_entropy_access`	boolean (default: False)	no	Whether to give the mount access to Stronghold's external entropy.
`local`	boolean (default: False)	no	Mark the mount as a local mount, which is not replicated and is unaffected by replication.
`options`	object	no	The options to pass into the backend. Should be a json object with string keys and values.
`plugin_name`	string	no	Name of the plugin to mount based from the name registered in the plugin catalog.
`plugin_version`	string	no	The semantic version of the plugin to use.
`seal_wrap`	boolean (default: False)	no	Whether to turn on seal wrapping for the mount.
`type`	string	no	The type of the backend. Example: "passthrough"

Responses

204: OK

DELETE /sys/mounts/{path}

Operation ID: `mounts-disable-secrets-engine`

Disable the mount point specified at the given path.

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	The path to mount to. Example: “aws/east”

Responses

200: OK

GET /sys/mounts/{path}/tune

Operation ID: `mounts-read-tuning-information`

Tune backend configuration parameters for this mount.

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	The path to mount to. Example: “aws/east”

Responses

200: OK

Parameter	Type	Required	Description
`allowed_managed_keys`	array	no	
`allowed_response_headers`	array	no	A list of headers to whitelist and allow a plugin to set on responses.
`audit_non_hmac_request_keys`	array	no	
`audit_non_hmac_response_keys`	array	no	
`default_lease_ttl`	integer	no	The default lease TTL for this mount.
`description`	string	no	User-friendly description for this credential backend.
`external_entropy_access`	boolean	no	
`force_no_cache`	boolean	no	
`listing_visibility`	string	no	
`max_lease_ttl`	integer	no	The max lease TTL for this mount.
`options`	object	no	The options to pass into the backend. Should be a json object with string keys and values.
`passthrough_request_headers`	array	no	
`plugin_version`	string	no	The semantic version of the plugin to use.
`replication_config`	object	no	KV replication configuration.
`token_type`	string	no	The type of token to issue (service or batch).
`user_lockout_counter_reset_duration`	integer	no	
`user_lockout_disable`	boolean	no	
`user_lockout_duration`	integer	no	
`user_lockout_threshold`	integer	no	

POST /sys/mounts/{path}/tune

Operation ID: `mounts-tune-configuration-parameters`

Tune backend configuration parameters for this mount.

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	The path to mount to. Example: “aws/east”

Request body parameters

Parameter	Type	Required	Description
`allowed_managed_keys`	array	no	
`allowed_response_headers`	array	no	A list of headers to whitelist and allow a plugin to set on responses.
`audit_non_hmac_request_keys`	array	no	The list of keys in the request data object that will not be HMAC’ed by audit devices.
`audit_non_hmac_response_keys`	array	no	The list of keys in the response data object that will not be HMAC’ed by audit devices.
`cmd_enable_repl`	boolean	no	Enable or disable the replication for this mount.
`default_lease_ttl`	string	no	The default lease TTL for this mount.
`description`	string	no	User-friendly description for this credential backend.
`listing_visibility`	string	no	Determines the visibility of the mount in the UI-specific listing endpoint. Accepted value are ‘unauth’ and ‘hidden’, with the empty default (’’) behaving like ‘hidden’.
`max_lease_ttl`	string	no	The max lease TTL for this mount.
`options`	object	no	The options to pass into the backend. Should be a json object with string keys and values.
`passthrough_request_headers`	array	no	A list of headers to whitelist and pass from the request to the plugin.
`plugin_version`	string	no	The semantic version of the plugin to use.
`src_address`	string	no	Address of the source Stronghold cluster for replication.
`src_ca_cert`	string	no	CA certificate for TLS connection to the source cluster.

Parameter	Type	Required	Description
`src_mount_path`	string	no	Mount path of the KV engine on the source cluster.
`src_secret_path`	array	no	List of secret paths to replicate from the source cluster.
`src_token`	string	no	Token for accessing the API of the remote server. Mutually exclusive with src_wrapping_token.
`src_wrapping_token`	string	no	Wrapping token that will be unwrapped on the remote server to obtain the replication token. Mutually exclusive with src_token.
`sync_period_min`	integer	no	Replication sync period in minutes.
`token_type`	string	no	The type of token to issue (service or batch).
`user_lockout_config`	object	no	The user lockout configuration to pass into the backend. Should be a json object with string keys and values.

Responses

200: OK

GET /sys/namespaces/

Operation ID: `namespaces-list-namespaces`

Create a new namespace at a new path.

Parameters

Parameter	Type	Location	Required	Description
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

POST /sys/namespaces/api-lock/lock

Operation ID: `namespaces-lock-namespace-api`

Lock the API for a namespace and all its descendants.

Request body parameters

Parameter	Type	Required	Description
`path`	string	no	Optional child namespace path to lock. If not provided, locks the current namespace.

Responses

200: OK

POST /sys/namespaces/api-lock/lock/{path}

Operation ID: `namespaces-lock-namespaces-api-lock-lock-path`

Lock the API for a namespace and all its descendants.

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	Optional child namespace path to lock. If not provided, locks the current namespace.

Responses

200: OK

POST /sys/namespaces/api-lock/unlock

Operation ID: `namespaces-unlock-namespace-api`

Unlock the API for a namespace and all its descendants.

Request body parameters

Parameter	Type	Required	Description
`path`	string	no	Optional child namespace path to unlock. If not provided, unlocks the current namespace.
`unlock_key`	string	no	The unlock key returned when the namespace was locked. Required unless using a root token.

Responses

200: OK

POST /sys/namespaces/api-lock/unlock/{path}

Operation ID: `namespaces-unlock-namespaces-api-lock-unlock-path`

Unlock the API for a namespace and all its descendants.

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	Optional child namespace path to unlock. If not provided, unlocks the current namespace.

Request body parameters

Parameter	Type	Required	Description
`unlock_key`	string	no	The unlock key returned when the namespace was locked. Required unless using a root token.

Responses

200: OK

GET /sys/namespaces/{path}

Operation ID: `namespaces-read-namespace`

Read namespace info

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	Path of the namespace.

Responses

200: OK

POST /sys/namespaces/{path}

Operation ID: `namespaces-create-namespace`

Create a new namespace at the given path.

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	Path of the namespace.

Request body parameters

Parameter	Type	Required	Description
`custom_metadata`	object	no	User-provided key-value pairs that are used to describe information about a secret.

Responses

204: OK

DELETE /sys/namespaces/{path}

Operation ID: `namespaces-delete-namespace`

Delete namespace specified at the given path.

Parameters

Parameter	Type	Location	Required	Description
`path`	string	path	yes	Path of the namespace.

Responses

200: OK

GET /sys/plugins/catalog

Operation ID: `plugins-catalog-list-plugins`

Lists all the plugins known to Stronghold

Responses

200: OK

Parameter	Type	Required	Description
`detailed`	object	no	

GET /sys/plugins/catalog/{name}

Operation ID: `plugins-catalog-read-plugin-configuration`

Return the configuration data for the plugin with the given name.

Required sudo: yes

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	The name of the plugin

Responses

200: OK

Parameter	Type	Required	Description
`args`	array	no	The args passed to plugin command.
`builtin`	boolean	no	
`command`	string	no	The command used to start the plugin. The executable defined in this command must exist in stronghold's plugin directory.
`deprecation_status`	string	no	
`name`	string	no	The name of the plugin
`sha256`	string	no	The SHA256 sum of the executable used in the command field. This should be HEX encoded.
`version`	string	no	The semantic version of the plugin to use.

POST /sys/plugins/catalog/{name}

Operation ID: `plugins-catalog-register-plugin`

Register a new plugin, or updates an existing one with the supplied name.

Required sudo: yes

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	The name of the plugin

Request body parameters

Parameter	Type	Required	Description
`args`	array	no	The args passed to plugin command.
`command`	string	no	The command used to start the plugin. The executable defined in this command must exist in stronghold's plugin directory.
`env`	array	no	The environment variables passed to plugin command. Each entry is of the form "key=value".
`sha256`	string	no	The SHA256 sum of the executable used in the command field. This should be HEX encoded.
`type`	string	no	The type of the plugin, may be auth, secret, or database
`version`	string	no	The semantic version of the plugin to use.

Responses

200: OK

DELETE /sys/plugins/catalog/{name}

Operation ID: `plugins-catalog-remove-plugin`

Remove the plugin with the given name.

Required sudo: yes

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	The name of the plugin

Responses

200: OK

GET /sys/plugins/catalog/{type}

Operation ID: `plugins-catalog-list-plugins-with-type`

List the plugins in the catalog.

Required sudo: yes

Parameters

Parameter	Type	Location	Required	Description
`type`	string	path	yes	The type of the plugin, may be auth, secret, or database
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

Parameter	Type	Required	Description
`keys`	array	no	List of plugin names in the catalog

GET /sys/plugins/catalog/{type}/{name}

Operation ID: `plugins-catalog-read-plugin-configuration-with-type`

Return the configuration data for the plugin with the given name.

Required sudo: yes

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	The name of the plugin
`type`	string	path	yes	The type of the plugin, may be auth, secret, or database

Responses

200: OK

Parameter	Type	Required	Description
`args`	array	no	The args passed to plugin command.
`builtin`	boolean	no	
`command`	string	no	The command used to start the plugin. The executable defined in this command must exist in stronghold's plugin directory.
`deprecation_status`	string	no	
`name`	string	no	The name of the plugin
`sha256`	string	no	The SHA256 sum of the executable used in the command field. This should be HEX encoded.
`version`	string	no	The semantic version of the plugin to use.

POST /sys/plugins/catalog/{type}/{name}

Operation ID: `plugins-catalog-register-plugin-with-type`

Register a new plugin, or updates an existing one with the supplied name.

Required sudo: yes

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	The name of the plugin
`type`	string	path	yes	The type of the plugin, may be auth, secret, or database

Request body parameters

Parameter	Type	Required	Description
`args`	array	no	The args passed to plugin command.
`command`	string	no	The command used to start the plugin. The executable defined in this command must exist in stronghold's plugin directory.
`env`	array	no	The environment variables passed to plugin command. Each entry is of the form "key=value".
`sha256`	string	no	The SHA256 sum of the executable used in the command field. This should be HEX encoded.
`version`	string	no	The semantic version of the plugin to use.

Responses

200: OK

DELETE /sys/plugins/catalog/{type}/{name}

Operation ID: `plugins-catalog-remove-plugin-with-type`

Remove the plugin with the given name.

Required sudo: yes

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	The name of the plugin
`type`	string	path	yes	The type of the plugin, may be auth, secret, or database

Responses

200: OK

POST /sys/plugins/reload/backend

Operation ID: `plugins-reload-backends`

Reload mounted plugin backends.

Request body parameters

Parameter	Type	Required	Description
`mounts`	array	no	The mount paths of the plugin backends to reload.
`plugin`	string	no	The name of the plugin to reload, as registered in the plugin catalog.
`scope`	string	no	

Responses

200: OK

Parameter	Type	Required	Description
`reload_id`	string	no	

202: OK

Parameter	Type	Required	Description
`reload_id`	string	no	

GET /sys/plugins/reload/backend/status

Operation ID: `enterprise-stub-read-plugins-reload-backend-status`

Responses

200: OK

GET /sys/policies/acl

Operation ID: `policies-list-acl-policies`

List the configured access control policies.

Parameters

Parameter	Type	Location	Required	Description
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

Parameter	Type	Required	Description
`keys`	array	no	
`policies`	array	no	

GET /sys/policies/acl/{name}

Operation ID: `policies-read-acl-policy`

Retrieve information about the named ACL policy.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	The name of the policy. Example: “ops”

Responses

200: OK

Parameter	Type	Required	Description
`name`	string	no	
`policy`	string	no	
`rules`	string	no	

POST /sys/policies/acl/{name}

Operation ID: `policies-write-acl-policy`

Add a new or update an existing ACL policy.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	The name of the policy. Example: “ops”

Request body parameters

Parameter	Type	Required	Description
`policy`	string	no	The rules of the policy.

Responses

204: OK

DELETE /sys/policies/acl/{name}

Operation ID: `policies-delete-acl-policy`

Delete the ACL policy with the given name.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	The name of the policy. Example: “ops”

Responses

204: OK

GET /sys/policies/egp

Operation ID: `enterprise-stub-list-policies-egp`

Parameters

Parameter	Type	Location	Required	Description
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /sys/policies/egp/{name}

Operation ID: `enterprise-stub-read-policies-egp-name`

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	

Responses

200: OK

POST /sys/policies/egp/{name}

Operation ID: `enterprise-stub-write-policies-egp-name`

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	

Responses

200: OK

DELETE /sys/policies/egp/{name}

Operation ID: `enterprise-stub-delete-policies-egp-name`

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	

Responses

204: empty body

GET /sys/policies/password

Operation ID: `policies-list-password-policies`

List the existing password policies.

Parameters

Parameter	Type	Location	Required	Description
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

Parameter	Type	Required	Description
`keys`	array	no	

GET /sys/policies/password/{name}

Operation ID: `policies-read-password-policy`

Retrieve an existing password policy.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	The name of the password policy.

Responses

204: OK

Parameter	Type	Required	Description
`policy`	string	no	

POST /sys/policies/password/{name}

Operation ID: `policies-write-password-policy`

Add a new or update an existing password policy.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	The name of the password policy.

Request body parameters

Parameter	Type	Required	Description
`policy`	string	no	The password policy

Responses

204: OK

DELETE /sys/policies/password/{name}

Operation ID: `policies-delete-password-policy`

Delete a password policy.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	The name of the password policy.

Responses

204: OK

GET /sys/policies/password/{name}/generate

Operation ID: `policies-generate-password-from-password-policy`

Generate a password from an existing password policy.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	The name of the password policy.

Responses

200: OK

Parameter	Type	Required	Description
`password`	string	no	

GET /sys/policies/rgp

Operation ID: `enterprise-stub-list-policies-rgp`

Parameters

Parameter	Type	Location	Required	Description
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /sys/policies/rgp/{name}

Operation ID: `enterprise-stub-read-policies-rgp-name`

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	

Responses

200: OK

POST /sys/policies/rgp/{name}

Operation ID: `enterprise-stub-write-policies-rgp-name`

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	

Responses

200: OK

DELETE /sys/policies/rgp/{name}

Operation ID: `enterprise-stub-delete-policies-rgp-name`

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	

Responses

204: empty body

GET /sys/policy

Operation ID: `policies-list`

List the configured access control policies.

Parameters

Parameter	Type	Location	Required	Description
`list`	string	query	no	Return a list if `true`

Responses

200: OK

Parameter	Type	Required	Description
`keys`	array	no	
`policies`	array	no	

GET /sys/policy/{name}

Operation ID: `policies-read-acl-policy2`

Retrieve the policy body for the named policy.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	The name of the policy. Example: “ops”

Responses

200: OK

Parameter	Type	Required	Description
`name`	string	no	
`policy`	string	no	
`rules`	string	no	

POST /sys/policy/{name}

Operation ID: `policies-write-acl-policy2`

Add a new or update an existing policy.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	The name of the policy. Example: “ops”

Request body parameters

Parameter	Type	Required	Description
`policy`	string	no	The rules of the policy.
`rules`	string	no	⚠ Deprecated. The rules of the policy.

Responses

204: OK

DELETE /sys/policy/{name}

Operation ID: `policies-delete-acl-policy2`

Delete the policy with the given name.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	The name of the policy. Example: “ops”

Responses

204: OK

GET /sys/pprof

Operation ID: `pprof-index`

Returns an HTML page listing the available profiles.

Responses

200: OK

GET /sys/pprof/allocs

Operation ID: `pprof-memory-allocations`

Returns a sampling of all past memory allocations.

Responses

200: OK

GET /sys/pprof/block

Operation ID: `pprof-blocking`

Returns stack traces that led to blocking on synchronization primitives

Responses

200: OK

GET /sys/pprof/cmdline

Operation ID: `pprof-command-line`

Returns the running program’s command line.

Responses

200: OK

GET /sys/pprof/goroutine

Operation ID: `pprof-goroutines`

Returns stack traces of all current goroutines.

Responses

200: OK

GET /sys/pprof/heap

Operation ID: `pprof-memory-allocations-live`

Returns a sampling of memory allocations of live object.

Responses

200: OK

GET /sys/pprof/mutex

Operation ID: `pprof-mutexes`

Returns stack traces of holders of contended mutexes

Responses

200: OK

GET /sys/pprof/profile

Operation ID: `pprof-cpu-profile`

Returns a pprof-formatted cpu profile payload.

Responses

200: OK

GET /sys/pprof/symbol

Operation ID: `pprof-symbols`

Returns the program counters listed in the request.

Responses

200: OK

GET /sys/pprof/threadcreate

Operation ID: `pprof-thread-creations`

Returns stack traces that led to the creation of new OS threads

Responses

200: OK

GET /sys/pprof/trace

Operation ID: `pprof-execution-trace`

Returns the execution trace in binary form.

Responses

200: OK

GET /sys/quotas/config

Operation ID: `rate-limit-quotas-read-configuration`

Create, update and read the quota configuration.

Responses

200: OK

Parameter	Type	Required	Description
`enable_rate_limit_audit_logging`	boolean	no	
`enable_rate_limit_response_headers`	boolean	no	
`rate_limit_exempt_paths`	array	no	

POST /sys/quotas/config

Operation ID: `rate-limit-quotas-configure`

Create, update and read the quota configuration.

Request body parameters

Parameter	Type	Required	Description
`enable_rate_limit_audit_logging`	boolean	no	If set, starts audit logging of requests that get rejected due to rate limit quota rule violations.
`enable_rate_limit_response_headers`	boolean	no	If set, additional rate limit quota HTTP headers will be added to responses.
`rate_limit_exempt_paths`	array	no	Specifies the list of exempt paths from all rate limit quotas. If empty no paths will be exempt.

Responses

204: OK

GET /sys/quotas/lease-count

Operation ID: `enterprise-stub-list-quotas-lease-count`

Parameters

Parameter	Type	Location	Required	Description
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /sys/quotas/lease-count/{name}

Operation ID: `enterprise-stub-read-quotas-lease-count-name`

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	

Responses

200: OK

POST /sys/quotas/lease-count/{name}

Operation ID: `enterprise-stub-write-quotas-lease-count-name`

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	

Responses

200: OK

DELETE /sys/quotas/lease-count/{name}

Operation ID: `enterprise-stub-delete-quotas-lease-count-name`

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	

Responses

204: empty body

GET /sys/quotas/rate-limit

Operation ID: `rate-limit-quotas-list`

Lists the names of all the rate limit quotas.

Parameters

Parameter	Type	Location	Required	Description
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

Parameter	Type	Required	Description
`keys`	array	no	

GET /sys/quotas/rate-limit/{name}

Operation ID: `rate-limit-quotas-read`

Get, create or update rate limit resource quota for an optional namespace or mount.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the quota rule.

Responses

200: OK

Parameter	Type	Required	Description
`block_interval`	integer	no	
`interval`	integer	no	
`name`	string	no	
`path`	string	no	
`rate`	number	no	
`role`	string	no	
`type`	string	no	

POST /sys/quotas/rate-limit/{name}

Operation ID: `rate-limit-quotas-write`

Get, create or update rate limit resource quota for an optional namespace or mount.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the quota rule.

Request body parameters

Parameter	Type	Required	Description
`block_interval`	integer	no	If set, when a client reaches a rate limit threshold, the client will be prohibited from any further requests until after the ‘block_interval’ has elapsed.
`interval`	integer	no	The duration to enforce rate limiting for (default ‘1s’).
`path`	string	no	Path of the mount or namespace to apply the quota. A blank path configures a global quota. For example namespace1/ adds a quota to a full namespace, namespace1/auth/userpass adds a quota to userpass in namespace1.
`rate`	number	no	The maximum number of requests in a given interval to be allowed by the quota rule. The ‘rate’ must be positive.
`role`	string	no	Login role to apply this quota to. Note that when set, path must be configured to a valid auth method with a concept of roles.
`type`	string	no	Type of the quota rule.

Responses

204: No Content

DELETE /sys/quotas/rate-limit/{name}

Operation ID: `rate-limit-quotas-delete`

Get, create or update rate limit resource quota for an optional namespace or mount.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the quota rule.

Responses

204: OK

GET /sys/rekey/backup

Operation ID: `rekey-read-backup-key`

Return the backup copy of PGP-encrypted unseal keys.

Responses

200: OK

Parameter	Type	Required	Description
`keys`	object	no	
`keys_base64`	object	no	
`nonce`	string	no	

DELETE /sys/rekey/backup

Operation ID: `rekey-delete-backup-key`

Delete the backup copy of PGP-encrypted unseal keys.

Responses

204: OK

GET /sys/rekey/init

Operation ID: `rekey-attempt-read-progress`

Reads the configuration and progress of the current rekey attempt.

Available without authentication: yes

Responses

200: OK

Parameter	Type	Required	Description
`backup`	boolean	no	
`n`	integer	no	
`nounce`	string	no	
`pgp_fingerprints`	array	no	
`progress`	integer	no	
`required`	integer	no	
`started`	string	no	
`t`	integer	no	
`verification_nonce`	string	no	
`verification_required`	boolean	no	

POST /sys/rekey/init

Operation ID: `rekey-attempt-initialize`

Initializes a new rekey attempt.

Available without authentication: yes

Request body parameters

Parameter	Type	Required	Description
`backup`	boolean	no	Specifies if using PGP-encrypted keys, whether Stronghold should also store a plaintext backup of the PGP-encrypted keys.
`pgp_keys`	array	no	Specifies an array of PGP public keys used to encrypt the output unseal keys. Ordering is preserved. The keys must be base64-encoded from their original binary representation. The size of this array must be the same as secret_shares.
`require_verification`	boolean	no	Turns on verification functionality
`secret_shares`	integer	no	Specifies the number of shares to split the unseal key into.
`secret_threshold`	integer	no	Specifies the number of shares required to reconstruct the unseal key. This must be less than or equal secret_shares. If using Stronghold HSM with auto-unsealing, this value must be the same as secret_shares.

Responses

200: OK

Parameter	Type	Required	Description
`backup`	boolean	no	
`n`	integer	no	
`nounce`	string	no	
`pgp_fingerprints`	array	no	
`progress`	integer	no	
`required`	integer	no	
`started`	string	no	
`t`	integer	no	
`verification_nonce`	string	no	
`verification_required`	boolean	no	

DELETE /sys/rekey/init

Operation ID: `rekey-attempt-cancel`

Cancels any in-progress rekey.

Available without authentication: yes

Responses

200: OK

GET /sys/rekey/recovery-key-backup

Operation ID: `rekey-read-backup-recovery-key`

Allows fetching or deleting the backup of the rotated unseal keys.

Responses

200: OK

Parameter	Type	Required	Description
`keys`	object	no	
`keys_base64`	object	no	
`nonce`	string	no	

DELETE /sys/rekey/recovery-key-backup

Operation ID: `rekey-delete-backup-recovery-key`

Allows fetching or deleting the backup of the rotated unseal keys.

Responses

204: OK

POST /sys/rekey/update

Operation ID: `rekey-attempt-update`

Enter a single unseal key share to progress the rekey of the Stronghold.

Available without authentication: yes

Request body parameters

Parameter	Type	Required	Description
`key`	string	no	Specifies a single unseal key share.
`nonce`	string	no	Specifies the nonce of the rekey attempt.

Responses

200: OK

Parameter	Type	Required	Description
`backup`	boolean	no	
`complete`	boolean	no	
`keys`	array	no	
`keys_base64`	array	no	
`n`	integer	no	
`nounce`	string	no	
`pgp_fingerprints`	array	no	
`progress`	integer	no	
`required`	integer	no	
`started`	string	no	
`t`	integer	no	
`verification_nonce`	string	no	
`verification_required`	boolean	no	

GET /sys/rekey/verify

Operation ID: `rekey-verification-read-progress`

Read the configuration and progress of the current rekey verification attempt.

Available without authentication: yes

Responses

200: OK

Parameter	Type	Required	Description
`n`	integer	no	
`nounce`	string	no	
`progress`	integer	no	
`started`	string	no	
`t`	integer	no	

POST /sys/rekey/verify

Operation ID: `rekey-verification-update`

Enter a single new key share to progress the rekey verification operation.

Available without authentication: yes

Request body parameters

Parameter	Type	Required	Description
`key`	string	no	Specifies a single unseal share key from the new set of shares.
`nonce`	string	no	Specifies the nonce of the rekey verification operation.

Responses

200: OK

Parameter	Type	Required	Description
`complete`	boolean	no	
`nounce`	string	no	

DELETE /sys/rekey/verify

Operation ID: `rekey-verification-cancel`

Cancel any in-progress rekey verification operation.

Available without authentication: yes

Responses

200: OK

Parameter	Type	Required	Description
`n`	integer	no	
`nounce`	string	no	
`progress`	integer	no	
`started`	string	no	
`t`	integer	no	

POST /sys/remount

Operation ID: `remount`

Initiate a mount migration

Required sudo: yes

Request body parameters

Parameter	Type	Required	Description
`from`	string	no	The previous mount point.
`to`	string	no	The new mount point.

Responses

200: OK

Parameter	Type	Required	Description
`migration_id`	string	no	

GET /sys/remount/status/{migration_id}

Operation ID: `remount-status`

Check status of a mount migration

Parameters

Parameter	Type	Location	Required	Description
`migration_id`	string	path	yes	The ID of the migration operation

Responses

200: OK

Parameter	Type	Required	Description
`migration_id`	string	no	
`migration_info`	object	no	

POST /sys/renew

Operation ID: `leases-renew-lease2`

Renews a lease, requesting to extend the lease.

Request body parameters

Parameter	Type	Required	Description
`increment`	integer	no	The desired increment in seconds to the lease
`lease_id`	string	no	The lease identifier to renew. This is included with a lease.
`url_lease_id`	string	no	The lease identifier to renew. This is included with a lease.

Responses

204: OK

POST /sys/renew/{url_lease_id}

Operation ID: `leases-renew-lease-with-id2`

Renews a lease, requesting to extend the lease.

Parameters

Parameter	Type	Location	Required	Description
`url_lease_id`	string	path	yes	The lease identifier to renew. This is included with a lease.

Request body parameters

Parameter	Type	Required	Description
`increment`	integer	no	The desired increment in seconds to the lease
`lease_id`	string	no	The lease identifier to renew. This is included with a lease.

Responses

204: OK

GET /sys/replication/status

Operation ID: `system-read-replication-status`

Available without authentication: yes

Responses

200: OK

POST /sys/revoke

Operation ID: `leases-revoke-lease2`

Revokes a lease immediately.

Request body parameters

Parameter	Type	Required	Description
`lease_id`	string	no	The lease identifier to renew. This is included with a lease.
`sync`	boolean (default: True)	no	Whether or not to perform the revocation synchronously
`url_lease_id`	string	no	The lease identifier to renew. This is included with a lease.

Responses

204: OK

POST /sys/revoke-force/{prefix}

Operation ID: `leases-force-revoke-lease-with-prefix2`

Revokes all secrets or tokens generated under a given prefix immediately

Required sudo: yes

Parameters

Parameter	Type	Location	Required	Description
`prefix`	string	path	yes	The path to revoke keys under. Example: “prod/aws/ops”

Responses

204: OK

POST /sys/revoke-prefix/{prefix}

Operation ID: `leases-revoke-lease-with-prefix2`

Revokes all secrets (via a lease ID prefix) or tokens (via the tokens’ path property) generated under a given prefix immediately.

Required sudo: yes

Parameters

Parameter	Type	Location	Required	Description
`prefix`	string	path	yes	The path to revoke keys under. Example: “prod/aws/ops”

Request body parameters

Parameter	Type	Required	Description
`sync`	boolean (default: True)	no	Whether or not to perform the revocation synchronously

Responses

204: OK

POST /sys/revoke/{url_lease_id}

Operation ID: `leases-revoke-lease-with-id2`

Revokes a lease immediately.

Parameters

Parameter	Type	Location	Required	Description
`url_lease_id`	string	path	yes	The lease identifier to renew. This is included with a lease.

Request body parameters

Parameter	Type	Required	Description
`lease_id`	string	no	The lease identifier to renew. This is included with a lease.
`sync`	boolean (default: True)	no	Whether or not to perform the revocation synchronously

Responses

204: OK

POST /sys/rotate

Operation ID: `encryption-key-rotate`

Rotates the backend encryption key used to persist data.

Required sudo: yes

Responses

204: OK

GET /sys/rotate/config

Operation ID: `encryption-key-read-rotation-configuration`

Configures settings related to the backend encryption key management.

Responses

200: OK

Parameter	Type	Required	Description
`enabled`	boolean	no	
`interval`	integer	no	
`max_operations`	integer	no	

POST /sys/rotate/config

Operation ID: `encryption-key-configure-rotation`

Configures settings related to the backend encryption key management.

Request body parameters

Parameter	Type	Required	Description
`enabled`	boolean	no	Whether automatic rotation is enabled.
`interval`	integer	no	How long after installation of an active key term that the key will be automatically rotated.
`max_operations`	integer	no	The number of encryption operations performed before the barrier key is automatically rotated.

Responses

204: OK

POST /sys/seal

Operation ID: `seal`

Seal the Stronghold.

Required sudo: yes

Responses

204: OK

GET /sys/seal-status

Operation ID: `seal-status`

Check the seal status of a Stronghold.

Available without authentication: yes

Responses

200:

Parameter	Type	Required	Description
`build_date`	string	no	
`cluster_id`	string	no	
`cluster_name`	string	no	
`hcp_link_resource_ID`	string	no	
`hcp_link_status`	string	no	
`initialized`	boolean	no	
`migration`	boolean	no	
`n`	integer	no	
`nonce`	string	no	
`progress`	integer	no	
`recovery_seal`	boolean	no	
`sealed`	boolean	no	
`storage_type`	string	no	
`t`	integer	no	
`type`	string	no	
`version`	string	no	

GET /sys/sealwrap/rewrap

Operation ID: `system-read-sealwrap-rewrap`

Responses

200: OK

POST /sys/sealwrap/rewrap

Operation ID: `system-write-sealwrap-rewrap`

Responses

200: OK

POST /sys/step-down

Operation ID: `step-down-leader`

Cause the node to give up active status.

Required sudo: yes

Responses

204: empty body

GET /sys/storage/raft/autopilot/configuration

Operation ID: `system-read-storage-raft-autopilot-configuration`

Returns autopilot configuration.

Responses

200: OK

POST /sys/storage/raft/autopilot/configuration

Operation ID: `system-write-storage-raft-autopilot-configuration`

Returns autopilot configuration.

Request body parameters

Parameter	Type	Required	Description
`cleanup_dead_servers`	boolean	no	Controls whether to remove dead servers from the Raft peer list periodically or when a new server joins.
`dead_server_last_contact_threshold`	integer	no	Limit on the amount of time a server can go without leader contact before being considered failed. This takes effect only when cleanup_dead_servers is set.
`disable_upgrade_migration`	boolean	no	Whether or not to perform automated version upgrades.
`dr_operation_token`	string	no	DR operation token used to authorize this request (if a DR secondary node).
`last_contact_threshold`	integer	no	Limit on the amount of time a server can go without leader contact before being considered unhealthy.
`max_trailing_logs`	integer	no	Amount of entries in the Raft Log that a server can be behind before being considered unhealthy.
`min_quorum`	integer	no	Minimum number of servers allowed in a cluster before autopilot can prune dead servers. This should at least be 3.
`server_stabilization_time`	integer	no	Minimum amount of time a server must be in a stable, healthy state before it can be added to the cluster.

Responses

200: OK

GET /sys/storage/raft/autopilot/state

Operation ID: `system-read-storage-raft-autopilot-state`

Returns the state of the raft cluster under integrated storage as seen by autopilot.

Responses

200: OK

POST /sys/storage/raft/bootstrap/answer

Operation ID: `system-write-storage-raft-bootstrap-answer`

Accepts an answer from the peer to be joined to the fact cluster.

Available without authentication: yes

Request body parameters

Parameter	Type	Required	Description
`answer`	string	no	
`cluster_addr`	string	no	
`non_voter`	boolean	no	
`server_id`	string	no	

Responses

200: OK

POST /sys/storage/raft/bootstrap/challenge

Operation ID: `system-write-storage-raft-bootstrap-challenge`

Creates a challenge for the new peer to be joined to the raft cluster.

Available without authentication: yes

Request body parameters

Parameter	Type	Required	Description
`server_id`	string	no	

Responses

200: OK

GET /sys/storage/raft/configuration

Operation ID: `system-read-storage-raft-configuration`

Returns the configuration of the raft cluster.

Responses

200: OK

POST /sys/storage/raft/configuration

Operation ID: `system-write-storage-raft-configuration`

Returns the configuration of the raft cluster in a DR secondary cluster.

Request body parameters

Parameter	Type	Required	Description
`dr_operation_token`	string	no	DR operation token used to authorize this request (if a DR secondary node).

Responses

200: OK

POST /sys/storage/raft/demote

Operation ID: `system-write-storage-raft-demote`

Demotes a voter to a permanent non-voter.

Request body parameters

Parameter	Type	Required	Description
`server_id`	string	no	

Responses

200: OK

POST /sys/storage/raft/promote

Operation ID: `system-write-storage-raft-promote`

Promotes a permanent non-voter to a voter.

Request body parameters

Parameter	Type	Required	Description
`server_id`	string	no	

Responses

200: OK

POST /sys/storage/raft/remove-peer

Operation ID: `system-write-storage-raft-remove-peer`

Remove a peer from the raft cluster.

Request body parameters

Parameter	Type	Required	Description
`dr_operation_token`	string	no	DR operation token used to authorize this request (if a DR secondary node).
`server_id`	string	no	

Responses

200: OK

GET /sys/storage/raft/snapshot

Operation ID: `system-read-storage-raft-snapshot`

Returns a snapshot of the current state of vault.

Responses

200: OK

POST /sys/storage/raft/snapshot

Operation ID: `system-write-storage-raft-snapshot`

Installs the provided snapshot, returning the cluster to the state defined in it.

Responses

200: OK

GET /sys/storage/raft/snapshot-auto/config

Operation ID: `system-list-storage-raft-snapshot-auto-config`

Lists all automatic snapshot configuration names.

Parameters

Parameter	Type	Location	Required	Description
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

GET /sys/storage/raft/snapshot-auto/config/{name}

Operation ID: `system-read-storage-raft-snapshot-auto-config-name`

Gets the configuration of the automatic snapshot.

Required sudo: yes

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the configuration to modify.

Responses

200: OK

POST /sys/storage/raft/snapshot-auto/config/{name}

Operation ID: `system-write-storage-raft-snapshot-auto-config-name`

Updates the configuration of the automatic snapshot.

Required sudo: yes

Creation supported: yes

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the configuration to modify.

Request body parameters

Parameter	Type	Required	Description
`aws_access_key_id`	string	no	S3 access key ID.
`aws_s3_bucket`	string	yes	S3 bucket to write snapshots to.
`aws_s3_ca_certificate`	string (default:)	no	S3 CA certificate PEM.
`aws_s3_disable_tls`	boolean (default: False)	no	Disable TLS for the S3 endpoint. This should only be used for testing purposes, typically in conjunction with `s3_endpoint`.
`aws_s3_endpoint`	string	no	S3 endpoint.
`aws_s3_region`	string (default:)	no	S3 region bucket is in.
`aws_secret_access_key`	string	no	S3 secret access key.
`file_prefix`	string (default: stronghold-snapshot)	no	Within the directory or bucket prefix given by `path_prefix`, the file or object name of snapshot files will start with this string.
`interval`	integer	yes	Time between snapshots. This can be either an integer number of seconds, or a Go duration format string (e.g. 24h).
`local_max_space`	integer (default: 0)	yes	For `storage_type=local`, the maximum space, in bytes, to use for all snapshots with the given `file_prefix` in the `path_prefix` directory. Snapshot attempts will fail if there is not enough space left in this allowance. Value `0` disables limit.
`path_prefix`	string	yes	For `storage_type=local`, the directory to write the snapshots in. For cloud storage types, the bucket prefix to use, also leading `/` is ignored. The trailing `/` is optional.
`retain`	integer (default: 3)	no	How many snapshots are to be kept; when writing a snapshot, if there are more

Parameter	Type	Required	Description
			snapshots already stored than this number, the oldest ones will be deleted.
`storage_type`	string (local, aws-s3)	yes	One of “local” or “s3”. The remaining parameters described below are all specific to the selected `storage_type` and prefixed accordingly.

Responses

200: OK

DELETE /sys/storage/raft/snapshot-auto/config/{name}

Operation ID: `system-delete-storage-raft-snapshot-auto-config-name`

Deletes the configuration of the automatic snapshot.

Required sudo: yes

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the configuration to modify.

Responses

204: empty body

GET /sys/storage/raft/snapshot-auto/status/{name}

Operation ID: `system-read-storage-raft-snapshot-auto-status-name`

Shows the status of the automatic snapshot.

Parameters

Parameter	Type	Location	Required	Description
`name`	string	path	yes	Name of the status to get.

Responses

200: OK

POST /sys/storage/raft/snapshot-force

Operation ID: `system-write-storage-raft-snapshot-force`

Installs the provided snapshot, returning the cluster to the state defined in it. This bypasses checks ensuring the current Autounseal or Shamir keys are consistent with the snapshot data.

Responses

200: OK

POST /sys/tools/hash

Operation ID: `generate-hash`

Generate a hash sum for input data

Request body parameters

Parameter	Type	Required	Description
`algorithm`	string (default: sha2-256)	no	Algorithm to use (POST body parameter). Valid values are: * sha2-224 * sha2-256 * sha2-384 * sha2-512 * streebog-256 * streebog-512 Defaults to “sha2-256”.
`format`	string (default: hex)	no	Encoding format to use. Can be “hex” or “base64”. Defaults to “hex”.
`input`	string	no	The base64-encoded input data
`urlalgorithm`	string	no	Algorithm to use (POST URL parameter)

Responses

200: OK

Parameter	Type	Required	Description
`sum`	string	no	

POST /sys/tools/hash/{urlalgorithm}

Operation ID: `generate-hash-with-algorithm`

Generate a hash sum for input data

Parameters

Parameter	Type	Location	Required	Description
`urlalgorithm`	string	path	yes	Algorithm to use (POST URL parameter)

Request body parameters

Parameter	Type	Required	Description
`algorithm`	string (default: sha2-256)	no	Algorithm to use (POST body parameter). Valid values are: * sha2-224 * sha2-256 * sha2-384 * sha2-512 * streebog-256 * streebog-512 Defaults to “sha2-256”.
`format`	string (default: hex)	no	Encoding format to use. Can be “hex” or “base64”. Defaults to “hex”.
`input`	string	no	The base64-encoded input data

Responses

200: OK

Parameter	Type	Required	Description
`sum`	string	no	

POST /sys/tools/random

Operation ID: `generate-random`

Generate random bytes

Request body parameters

Parameter	Type	Required	Description
`bytes`	integer (default: 32)	no	The number of bytes to generate (POST body parameter). Defaults to 32 (256 bits).
`format`	string (default: base64)	no	Encoding format to use. Can be “hex” or “base64”. Defaults to “base64”.
`source`	string (default: platform)	no	Which system to source random data from, ether “platform”, “seal”, or “all”.
`urlbytes`	string	no	The number of bytes to generate (POST URL parameter)

Responses

200: OK

Parameter	Type	Required	Description
`random_bytes`	string	no	

POST /sys/tools/random/{source}

Operation ID: `generate-random-with-source`

Generate random bytes

Parameters

Parameter	Type	Location	Required	Description
`source`	string	path	yes	Which system to source random data from, ether “platform”, “seal”, or “all”.

Request body parameters

Parameter	Type	Required	Description
`bytes`	integer (default: 32)	no	The number of bytes to generate (POST body parameter). Defaults to 32 (256 bits).
`format`	string (default: base64)	no	Encoding format to use. Can be “hex” or “base64”. Defaults to “base64”.
`urlbytes`	string	no	The number of bytes to generate (POST URL parameter)

Responses

200: OK

Parameter	Type	Required	Description
`random_bytes`	string	no	

POST /sys/tools/random/{source}/{urlbytes}

Operation ID: `generate-random-with-source-and-bytes`

Generate random bytes

Parameters

Parameter	Type	Location	Required	Description
`source`	string	path	yes	Which system to source random data from, ether “platform”, “seal”, or “all”.
`urlbytes`	string	path	yes	The number of bytes to generate (POST URL parameter)

Request body parameters

Parameter	Type	Required	Description
`bytes`	integer (default: 32)	no	The number of bytes to generate (POST body parameter). Defaults to 32 (256 bits).
`format`	string (default: base64)	no	Encoding format to use. Can be “hex” or “base64”. Defaults to “base64”.

Responses

200: OK

Parameter	Type	Required	Description
`random_bytes`	string	no	

POST /sys/tools/random/{urlbytes}

Operation ID: `generate-random-with-bytes`

Generate random bytes

Parameters

Parameter	Type	Location	Required	Description
`urlbytes`	string	path	yes	The number of bytes to generate (POST URL parameter)

Request body parameters

Parameter	Type	Required	Description
`bytes`	integer (default: 32)	no	The number of bytes to generate (POST body parameter). Defaults to 32 (256 bits).
`format`	string (default: base64)	no	Encoding format to use. Can be “hex” or “base64”. Defaults to “base64”.
`source`	string (default: platform)	no	Which system to source random data from, ether “platform”, “seal”, or “all”.

Responses

200: OK

Parameter	Type	Required	Description
`random_bytes`	string	no	

POST /sys/unseal

Operation ID: `unseal`

Unseal the Stronghold.

Available without authentication: yes

Request body parameters

Parameter	Type	Required	Description
`key`	string	no	Specifies a single unseal key share. This is required unless reset is true.
`reset`	boolean	no	Specifies if previously-provided unseal keys are discarded and the unseal process is reset.

Responses

200:

Parameter	Type	Required	Description
`build_date`	string	no	
`cluster_id`	string	no	
`cluster_name`	string	no	
`hcp_link_resource_ID`	string	no	
`hcp_link_status`	string	no	
`initialized`	boolean	no	
`migration`	boolean	no	
`n`	integer	no	
`nonce`	string	no	
`progress`	integer	no	
`recovery_seal`	boolean	no	
`sealed`	boolean	no	
`storage_type`	string	no	
`t`	integer	no	
`type`	string	no	
`version`	string	no	

GET /sys/version-history

Operation ID: `version-history`

Returns map of historical version change entries

Parameters

Parameter	Type	Location	Required	Description
`list`	string (true)	query	yes	Must be set to `true`

Responses

200: OK

Parameter	Type	Required	Description
`key_info`	object	no	
`keys`	array	no	

GET /sys/wrapping/lookup

Operation ID: `read-wrapping-properties2`

Look up wrapping properties for the requester’s token.

Available without authentication: yes

Responses

200: OK

Parameter	Type	Required	Description
`creation_path`	string	no	
`creation_time`	string	no	
`creation_ttl`	integer	no	

POST /sys/wrapping/lookup

Operation ID: `read-wrapping-properties`

Look up wrapping properties for the given token.

Available without authentication: yes

Request body parameters

Parameter	Type	Required	Description
`token`	string	no	

Responses

200: OK

Parameter	Type	Required	Description
`creation_path`	string	no	
`creation_time`	string	no	
`creation_ttl`	integer	no	

POST /sys/wrapping/rewrap

Operation ID: `rewrap`

Rotates a response-wrapped token.

Request body parameters

Parameter	Type	Required	Description
`token`	string	no	

Responses

200: OK

POST /sys/wrapping/unwrap

Operation ID: `unwrap`

Unwraps a response-wrapped token.

Request body parameters

Parameter	Type	Required	Description
`token`	string	no	

Responses

200: OK

204: No content

POST /sys/wrapping/wrap

Operation ID: `wrap`

Response-wraps an arbitrary JSON object.

Responses

200: OK

Console utilities

This section describes the console utilities for working with Deckhouse Stronghold.

FAQ

Release notes

v1.18 April 2026

Version `1.18` focuses on the development of external key management systems, `PKI`, and new audit capabilities.

- Managed Keys for working with key material in external trusted systems without storing private keys inside Stronghold. Supported secret engines: `Transit`, `PKI`, and `SSH`.
- Added support for `Yandex KMS` managed keys and keys using the `PKCS#11` standard.
- User authentication via an external `SAML 2.0` Identity Provider using the `Web SSO` profile.
- Added management of `KV` mount replication parameters in the web interface.
- Added support for single-element RDN in distinguished names within `PKI` for compatibility with OpenSSL / Microsoft CA.
- Added record filtering and field exclusion mechanisms for audit devices.
- Added snapshot inspection via the `stronghold operator raft snapshot inspect` command.
- Added ability to manage the `max_ttl` parameter for ACME certificates.
- Improved compatibility with Vault Enterprise Auto-Snapshots — snapshot configuration is preserved when migrating from Vault Enterprise.
- CVEs fixed: GHSA-jqcq-xjh3-6g23, CVE-2026-33186, CVE-2026-33487, CVE-2025-15558

v1.17 February 2026

Version `1.17` continued development of integrations, protection mechanisms, and enterprise operational scenarios.

- Added support for `WebAuthn` — passwordless authentication (FIDO2/Passkeys).
- Added support for external Stronghold plugins running in DKP.
- Added namespace locks and an interface for managing them.
- Added support for the `LDAP secrets engine` in the web interface.
- Added support for `Yandex KMS` as a seal backend.
- Expanded `Agent` operational scenarios.
- Added support for `raft` nodes in `non-voter` mode.
- Enhanced deployment scenarios on arbiter node groups and test cluster parameters.

v1.16 August 2025

Version `1.16` was a major functional release, introducing namespaces, new web interface features, replication mechanisms, cryptographic enhancements, and a new product edition.

- Added support for `Namespaces` in `EE`.
- Implemented multi-factor authentication (`MFA`) with support for `TOTP` and `Multifactor`.

- Introduced the Deckhouse Stronghold `CE` (`Community Edition`), available for free installation.
- Added web interface management for `OIDC` roles, `AppRole`, and password policies.
- Added replication metrics.
- Added `seal wrap` — an additional encryption mechanism for the most sensitive internal data on top of the standard Stronghold cryptographic barrier.
- Added `CryptoPro seal wrapper` for scenarios using Russian cryptography.
- The web interface received a more complete Russian localization and a dark theme.
- Added `ClickHouse` support and a web interface for working with it.
- Added `TLS 1.3` support with `Magma` and `Kuznechik` GOST encryption.
- Added support for `GOST 34.10-2012 X.509` certificates.
- Version `1.16` received FSTEC of Russia certificate No. 5038 dated February 10, 2026, for Deckhouse Stronghold software.

v1.15 February 2025

Version numbering changed, transitioning from v1.1 to v1.15. Version `1.15` focused on operational scenarios, backup, and the first significant interface improvements.

- Added scheduled `Raft snapshots` backup with storage in `S3` or file system and API management.
- Expanded `KV` replication capabilities.
- Improved the web interface.
- Added automatic unsealing via `HSM` / `PKCS#11`, including support for Rutoken ECP 3.0.

v1.1 March 2024

- Automatic unsealing with key storage in Stronghold node memory
- Russian language interface
- Included in the Russian Software Registry, Registry entry No. 22339 dated April 24, 2024
- Added integration with the platform secret delivery module secrets-store-integration

v1.0 February 2024 / Vault v1.14.x

- Deployment as a DKP module
- Integration with platform DEX authentication