hvac

Release 0.6.1

Sep 17, 2018

Contents:

Source code repository hosted at github.com/hvac/hvac.

CHAPTER 1

hvac

HashiCorp Vault API client for Python 2.7/3.x Tested against Vault v0.1.2 and HEAD. Requires v0.1.2 or later.

1.1 Documentation

Documentation for this module is hosted on readthedocs.io.

1.2 Getting started

1.2.1 Installation

```
pip install hvac
```

or

```
pip install "hvac[parser]"
```

if you would like to be able to return parsed HCL data as a Python dict for methods that support it.

1.2.2 Initialize the client

```
import os

import hvac

# Using plaintext
client = hvac.Client()
client = hvac.Client(url='http://localhost:8200')
```

(continues on next page)

(continued from previous page)

```
client = hvac.Client(url='http://localhost:8200', token=os.environ['VAULT_TOKEN'])

# Using TLS
client = hvac.Client(url='https://localhost:8200')

# Using TLS with client-side certificate authentication
client = hvac.Client(url='https://localhost:8200', cert=('path/to/cert.pem', 'path/to/
↪key.pem'))
```

1.2.3 Read and write to secret backends

```
client.write('secret/foo', baz='bar', lease='1h')

print(client.read('secret/foo'))

client.delete('secret/foo')
```

1.2.4 Authenticate using token auth backend

```
# Token
client.token = 'MY_TOKEN'
assert client.is_authenticated() # => True
```

CHAPTER 2

Usage

2.1 Secrets Engines

2.1.1 KV - Wrapper

The *hvac.api.secrets_engines.Kv* instance under the *Client class's kv attribute* is a wrapper to expose either version 1 (*KvV1*) or version 2 of the key/value secrets engines' API methods (*KvV2*). At present, this class defaults to version 2 when accessing methods on the instance.

Setting the Default KV Version

hvac.api.secrets_engines.KvV1.read_secret()

```
import hvac
client = hvac.Client()

client.kv.default_kv_version = 1
client.kv.read_secret(path='hvac')  # => calls hvac.api.secrets_engines.KvV1.read_
↪secret
```

Explicitly Calling a KV Version Method

hvac.api.secrets_engines.KvV1.list_secrets()

```
import hvac
client = hvac.Client()

client.kv.v1.read_secret(path='hvac')
client.kv.v2.read_secret_version(path='hvac')
```

Specific KV Version Usage

KV - Version 1

Note: Every method under the *Kv class's v1 attribute* includes a *mount_point* parameter that can be used to address the KvV1 secret engine under a custom mount path. E.g., If enabling the KvV1 secret engine using Vault's CLI commands via *vault secrets enable -path=my-kvv1 -version=1 kv*", the *mount_point* parameter in *hvac.api.secrets_engines.KvV1()* methods would be set to "my-kvv1".

Read a Secret

hvac.api.secrets_engines.KvV1.read_secret()

```
import hvac
client = hvac.Client()

# The following path corresponds, when combined with the mount point, to a full Vault␣
↪API route of "v1/secretz/hvac"
mount_point = 'secretz'
secret_path = 'hvac'

client.kv.v1.read_secret(
    path=secret_path,
    mount_point=mount_point,
)
print('The "psst" key under the secret path ("/v1/secret/hvac") is: {psst}'.format(
    psst=read_secret_result['data']['psst'],
))
```

List Secrets

hvac.api.secrets_engines.KvV1.list_secrets()

```
import hvac
client = hvac.Client()

list_secrets_result = client.kv.v1.list_secrets(path='hvac')

print('The following keys found under the selected path ("/v1/secret/hvac"): {keys}'.
↪format(
    keys=','.join(list_secrets_result['data']['keys']),
))
```

Create or Update a Secret

hvac.api.secrets_engines.KvV1.create_or_update_secret()

```
import hvac
client = hvac.Client()
hvac_secret = {
```

(continues on next page)

(continued from previous page)

```
    'psst': 'this is so secret yall',
}

client.kv.v1.create_or_update_secret(
    path='hvac',
    secret=hvac_secret,
)

read_secret_result = client.kv.v1.read_secret(
    path='hvac',
)
print('The "psst" key under the secret path ("/v1/secret/hvac") is: {psst}'.format(
    psst=read_secret_result['data']['psst'],
))
```

Delete a Secret

hvac.api.secrets_engines.KvV1.delete_secret()

```
import hvac
client = hvac.Client()

client.kv.v1.delete_secret(
    path='hvac',
)

# The following will raise a :py:class:`hvac.exceptions.InvalidPath` exception.
read_secret_result = client.kv.v1.read_secret(
    path='hvac',
)
```

KV - Version 2

Note: Every method under the *Kv class's v2 attribute* includes a *mount_point* parameter that can be used to address the KvV2 secret engine under a custom mount path. E.g., If enabling the KvV2 secret engine using Vault's CLI commands via *vault secrets enable -path=my-kvv2 -version=2 kv*", the *mount_point* parameter in *hvac.api.secrets_engines.KvV2()* methods would be set to "my-kvv2".

Configuration

hvac.api.secrets_engines.KvV2.configure()

Setting the default *max_versions* for a key/value engine version 2 under a path of *kv*:

```
import hvac
client = hvac.Client()

client.kv.v2.configure(
    max_versions=20,
```

(continues on next page)

(continued from previous page)

```
    mount_point='kv',
)
```

Setting the default *cas_required* (check-and-set required) flag under the implicit default path of *secret*:

```
import hvac
client = hvac.Client()

client.kv.v2.configure(
    cas_required=True,
)
```

Read Configuration

hvac.api.secrets_engines.KvV2.configure()

Reading the configuration of a KV version 2 engine mounted under a path of *kv*:

```
import hvac
client = hvac.Client()

kv_configuration = client.kv.v2.read_configuration(
    mount_point='kv',
)
print('Config under path "kv": max_versions set to "{max_ver}"'.format(
    max_ver=kv_configuration['data']['max_versions'],
))
print('Config under path "kv": check-and-set require flag set to {cas}'.format(
    cas=kv_configuration['data']['cas_required'],
))
```

Read Secret Versions

hvac.api.secrets_engines.KvV2.read_secret_version()

Read the latest version of a given secret/path ("hvac"):

```
import hvac
client = hvac.Client()

secret_version_response = client.kv.v2.read_secret_version(
    path='hvac',
)
print('Latest version of secret under path "hvac" contains the following keys: {data}.
↪format(
    data=secret_version_response['data']['data'].keys(),
))
print('Latest version of secret under path "hvac" created at: {date}.format(
    date=secret_version_response['data']['metadata']['created_time'],
))
print('Latest version of secret under path "hvac" is version #{ver}.format(
    ver=secret_version_response['data']['metadata']['version'],
))
```

Read specific version (1) of a given secret/path ("hvac"):

```
import hvac
client = hvac.Client()

secret_version_response = client.kv.v2.read_secret_version(
    path='hvac',
    version=1,
)
print('Version 1 of secret under path "hvac" contains the following keys: {data}.
↪format(
    data=secret_version_response['data']['data'].keys(),
))
print('Version 1 of secret under path "hvac" created at: {date}.format(
    date=secret_version_response['data']['metadata']['created_time'],
))
```

Create/Update Secret

hvac.api.secrets_engines.KvV2.create_or_update_secret()

```
import hvac
client = hvac.Client()

client.kv.v2.create_or_update_secret(
    path='hvac',
    secret=dict(pssst='this is secret'),
)
```

cas parameter with an argument that doesn't match the current version:

```
import hvac
client = hvac.Client()

# Assuming a current version of "6" for the path "hvac" =>
client.kv.v2.create_or_update_secret(
    path='hvac',
    secret=dict(pssst='this is secret'),
    cas=5,
)  # Raises hvac.exceptions.InvalidRequest
```

cas parameter set to *0* will only succeed if the path hasn't already been written.

```
import hvac
client = hvac.Client()

client.kv.v2.create_or_update_secret(
    path='hvac',
    secret=dict(pssst='this is secret #1'),
    cas=0,
)

client.kv.v2.create_or_update_secret(
    path='hvac',
    secret=dict(pssst='this is secret #2'),
    cas=0,
)  # => Raises hvac.exceptions.InvalidRequest
```

Delete Latest Version of Secret

hvac.api.secrets_engines.KvV2.delete_latest_version_of_secret()

```
import hvac
client = hvac.Client()

client.kv.v2.delete_latest_version_of_secret(
    path=hvac,
)
```

Delete Secret Versions

hvac.api.secrets_engines.KvV2.delete_secret_versions()

Marking the first 3 versions of a secret deleted under path "hvac":

```
import hvac
client = hvac.Client()

client.kv.v2.delete_secret_versions(
    path='hvac',
    versions=[1, 2, 3],
)
```

Undelete Secret Versions

hvac.api.secrets_engines.KvV2.undelete_secret_versions()

Marking the last 3 versions of a secret deleted under path "hvac" as "undeleted":

```
import hvac
client = hvac.Client()

hvac_path_metadata = client.kv.v2.read_secret_metadata(
    path='hvac',
)

oldest_version = hvac_path_metadata['data']['oldest_version']
current_version = hvac_path_metadata['data']['current_version']
versions_to_undelete = range(max(oldest_version, current_version - 2), current_
↪version + 1)

client.kv.v2.undelete_secret_versions(
    path='hvac',
    versions=versions_to_undelete,
)
```

Destroy Secret Versions

hvac.api.secrets_engines.KvV2.destroy_secret_versions()

Destroying the first three versions of a secret under path 'hvac':

```
import hvac
client = hvac.Client()

client.kv.v2.destroy_secret_versions(
    path='hvac',
    versions=[1, 2, 3],
)
```

List Secrets

hvac.api.secrets_engines.KvV2.list_secrets()

Listing secrets under the 'hvac' path prefix:

```
import hvac
client = hvac.Client()

client.kv.v2.create_or_update_secret(
    path='hvac/big-ole-secret',
    secret=dict(pssst='this is a large secret'),
)

client.kv.v2.create_or_update_secret(
    path='hvac/lil-secret',
    secret=dict(pssst='this secret... not so big'),
)

list_response = client.kv.v2.list_secrets(
    path='hvac',
)

print('The following paths are available under "hvac" prefix: {keys}'.format(
    keys=','.join(list_response['data']['keys']),
))
```

Read Secret Metadata

hvac.api.secrets_engines.KvV2.read_secret_metadata()

```
import hvac
client = hvac.Client()

hvac_path_metadata = client.kv.v2.read_secret_metadata(
    path='hvac',
)

print('Secret under path hvac is on version {cur_ver}, with an oldest version of {old_
↪ver}'.format(
    cur_ver=hvac_path_metadata['data']['oldest_version'],
    old_ver=hvac_path_metadata['data']['current_version'],
))
```

Update Metadata

hvac.api.secrets_engines.KvV2.update_metadata()

Set max versions for a given path ("hvac") to 3:

```
import hvac
client = hvac.Client()

client.kv.v2.update_metadata(
    path='hvac',
    max_versions=3,
)
```

Set cas (check-and-set) parameter as required for a given path ("hvac"):

```
import hvac
client = hvac.Client()

client.kv.v2.update_metadata(
    path='hvac',
    cas_required=True,
)
```

Delete Metadata and All Versions

hvac.api.secrets_engines.KvV2.delete_metadata_and_all_versions()

Delete all versions and metadata for a given path:

```
import hvac
client = hvac.Client()

client.kv.v2.delete_metadata_and_all_versions(
    path='hvac',
)
```

2.2 Auth Methods

2.2.1 Approle

Authentication

```
client.auth_approle('MY_ROLE_ID', 'MY_SECRET_ID')
```

2.2.2 AWS

Authentication

IAM authentication method:

```
client.auth_aws_iam('MY_AWS_ACCESS_KEY_ID', 'MY_AWS_SECRET_ACCESS_KEY')
client.auth_aws_iam('MY_AWS_ACCESS_KEY_ID', 'MY_AWS_SECRET_ACCESS_KEY', 'MY_AWS_
↪SESSION_TOKEN')
client.auth_aws_iam('MY_AWS_ACCESS_KEY_ID', 'MY_AWS_SECRET_ACCESS_KEY', role='MY_ROLE
↪')

import boto3
session = boto3.Session()
credentials = session.get_credentials()
client.auth_aws_iam(credentials.access_key, credentials.secret_key, credentials.token)
```

2.2.3 GCP

Authentication

```
# GCP (from GCE instance)
import requests

VAULT_ADDR="https://vault.example.com:8200"
ROLE="example"
AUDIENCE_URL =  VAULT_ADDR + "/vault/" + ROLE
METADATA_HEADERS = {'Metadata-Flavor': 'Google'}
FORMAT = 'full'

url = 'http://metadata.google.internal/computeMetadata/v1/instance/service-accounts/
↪default/identity?audience={}&format={}'.format(AUDIENCE_URL, FORMAT)
r = requests.get(url, headers=METADATA_HEADERS)
client.auth_gcp(ROLE, r.text)
```

2.2.4 GitHub

Note: Every method under the *Client class's github attribute* includes a *mount_point* parameter that can be used to address the Github auth method under a custom mount path. E.g., If enabling the Github auth method using Vault's CLI commands via *vault auth enable -path=my-github github*", the *mount_point* parameter in *hvac.api.auth.Github()* methods would be set to "my-github".

Enabling the Auth Method

hvac.v1.Client.enable_auth_backend()

```
import hvac
client = hvac.Client()

github_auth_path = 'company-github'
description = 'Auth method for use by team members in our company's Github␣
↪organization'

if '%s/' % github_auth_path not in vault_client.list_auth_backends():
    print('Enabling the github auth backend at mount_point: {path}'.format(
        path=github_auth_path,
```

(continues on next page)

(continued from previous page)

```
    ))
    client.enable_auth_backend(
        backend_type='github',
        description=description,
        mount_point=github_auth_path,
    )
```

Configure Connection Parameters

hvac.api.auth.Github.configure()

```
import hvac
client = hvac.Client()

client.github.configure(
    organization='our-lovely-company',
    max_ttl='48h',  # i.e., A given token can only be renewed for up to 48 hours
)
```

Reading Configuration

hvac.api.auth.Github.read_configuration()

```
import hvac
client = hvac.Client()

github_config = client.github.read_configuration()
print('The Github auth method is configured with a ttl of: {ttl}'.format(
    ttl=github_config['data']['ttl']
)
```

Mapping Teams to Policies

hvac.api.auth.Github.map_team()

```
import hvac
client = hvac.Client()

teams = [
    dict(name='some-dev-team', policies=['dev-team']),
    dict(name='admin-team', policies=['administrator']),
]
for team in teams:
    client.github.map_team(
        team_name=team['name'],
        policies=team['policies'],
    )
```

Reading Team Mappings

hvac.api.auth.Github.read_team_mapping()

```
import hvac
client = hvac.Client()

team_name = 'my-super-cool-team'
github_config = client.github.read_team_mapping(
    team_name=team_name,
)
print('The Github team {team} is mapped to the following policies: {policies}'.format(
    team=team_name,
    policies=github_config['data']['value'],
)
```

Mapping Users to Policies

hvac.api.auth.Github.map_user()

```
import hvac
client = hvac.Client()

users = [
    dict(name='some-dev-user', policies=['dev-team']),
    dict(name='some-admin-user', policies=['administrator']),
]
for user in users:
    client.github.map_user(
        user_name=user['name'],
        policies=user['policies'],
    )
```

Reading User Mappings

hvac.api.auth.Github.read_user_mapping()

```
import hvac
client = hvac.Client()

user_name = 'some-dev-user'
github_config = client.github.read_user_mapping(
    user_name=user_name,
)
print('The Github user "{user}" is mapped to the following policies: {policies}'.
↪format(
    user=user_name,
    policies=github_config['data']['value'],
)
```

Authentication / Login

hvac.api.auth.Github.login()

Log in and automatically update the underlying "token" attribute on the *hvac.adapters.Adapter()* instance:

```
import hvac
client = hvac.Client()
login_response = client.github.login(token='some personal github token')
```

2.2.5 Kubernetes

Authentication

```
# Kubernetes (from k8s pod)
f = open('/var/run/secrets/kubernetes.io/serviceaccount/token')
jwt = f.read()
client.auth_kubernetes("example", jwt)
```

2.2.6 LDAP

Note: Every method under the *Client class's ldap attribute* includes a *mount_point* parameter that can be used to address the LDAP auth method under a custom mount path. E.g., If enabling the LDAP auth method using Vault's CLI commands via *vault auth enable -path=my-ldap ldap*", the *mount_point* parameter in *hvac.api.auth.Ldap()* methods would be set to "my-ldap".

Enabling the LDAP Auth Method

hvac.v1.Client.enable_auth_backend()

```
import hvac
client = hvac.Client()

ldap_auth_path = 'company-ldap'
description = "Auth method for use by team members in our company's LDAP organization"

if '%s/' % ldap_auth_path not in vault_client.list_auth_backends():
    print('Enabling the ldap auth backend at mount_point: {path}'.format(
        path=ldap_auth_path,
    ))
    client.enable_auth_backend(
        backend_type='ldap',
        description=description,
        mount_point=ldap_auth_path,
    )
```

Configure LDAP Auth Method Settings

hvac.api.auth.Ldap.configure()

```
import hvac
client = hvac.Client()

client.ldap.configure(
```

(continues on next page)

(continued from previous page)

```
    user_dn='dc=users,dc=hvac,dc=network',
    group_dn='ou=groups,dc=hvac,dc=network',
    url='ldaps://ldap.hvac.network:12345',
    bind_dn='cn=admin,dc=hvac,dc=network',
    bind_pass='ourverygoodadminpassword'
    user_attr='uid',
    group_attr='cn',
)
```

Reading the LDAP Auth Method Configuration

hvac.api.auth.Ldap.read_configuration()

```
import hvac
client = hvac.Client()

ldap_configuration = client.ldap.read_configuration()
print('The LDAP auth method is configured with a LDAP server URL of: {url}'.format(
    url=ldap_configuration['data']['url']
)
```

Create or Update a LDAP Group Mapping

hvac.api.auth.Ldap.create_or_update_group()

```
import hvac
client = hvac.Client()

client.ldap.create_or_update_group(
    name='some-dudes',
    policies=['policy-for-some-dudes'],
)
```

List LDAP Group Mappings

hvac.api.auth.Ldap.list_groups()

```
import hvac
client = hvac.Client()

ldap_groups = client.ldap.list_groups()
print('The following groups are configured in the LDAP auth method: {groups}'.format(
    groups=','.join(ldap_groups['data']['keys'])
)
```

Read LDAP Group Mapping

hvac.api.auth.Ldap.read_group()

```
import hvac
client = hvac.Client()

some_dudes_ldap_group = client.ldap.read_group(
    name='somedudes',
)
print('The "somedudes" group in the LDAP auth method are mapped to the following␣
↪policies: {policies}'.format(
    policies=','.join(some_dudes_ldap_group['data']['policies'])
)
```

Deleting a LDAP Group Mapping

hvac.api.auth.Ldap.delete_group()

```
import hvac
client = hvac.Client()

client.ldap.delete_group(
    name='some-group',
)
```

Creating or Updating a LDAP User Mapping

hvac.api.auth.Ldap.create_or_update_user()

```
import hvac
client = hvac.Client()

client.ldap.create_or_update_user(
    username='somedude',
    policies=['policy-for-some-dudes'],
)
```

Listing LDAP User Mappings

hvac.api.auth.Ldap.list_users()

```
import hvac
client = hvac.Client()

ldap_users = client.ldap.list_users()
print('The following users are configured in the LDAP auth method: {users}'.format(
    users=','.join(ldap_users['data']['keys'])
)
```

Reading a LDAP User Mapping

hvac.api.auth.Ldap.read_user()

```
import hvac
client = hvac.Client()

some_dude_ldap_user = client.ldap.read_user(
    username='somedude'
)
print('The "somedude" user in the LDAP auth method is mapped to the following␣
↪policies: {policies}'.format(
    policies=','.join(some_dude_ldap_user['data']['policies'])
)
```

Deleting a Configured User Mapping

hvac.api.auth.Ldap.delete_user()

```
import hvac
client = hvac.Client()

client.ldap.delete_user(
    username='somedude',
)
```

Authentication / Login

`hvac.api.auth.Ldap.login_with_user()`

For a LDAP backend mounted under a non-default (ldap) path. E.g., via Vault CLI with *vault auth enable -path=prod-ldap ldap*

```
from getpass import getpass

import hvac

service_account_username = 'someuser'
password_prompt = 'Please enter your password for the LDAP authentication backend: '
service_account_password = getpass(prompt=password_prompt)

client = hvac.Client()

# Here the mount_point parameter corresponds to the path provided when enabling the␣
↪backend
client.ldap.login(
    username=service_account_username,
    password=service_account_password,
    mount_point='prod-ldap'
)
print(client.is_authenticated)  # => True
```

2.2.7 MFA

Configure MFA Auth Method Settings

hvac.api.auth.Mfa.configure()

Note: The legacy/unsupported MFA auth method covered by this class's configuration API route only supports integration with a subset of Vault auth methods. See the list of supported auth methods in this module's `"SUPPORTED_AUTH_METHODS" attribute` and/or the associated Vault MFA documentation for additional information.

```
import hvac
client = hvac.Client()

userpass_auth_path = 'some-userpass'

if '%s/' % userpass_auth_path not in vault_client.list_auth_backends():
    print('Enabling the userpass auth backend at mount_point: {path}'.format(
        path=userpass_auth_path,
    ))
    client.enable_auth_backend(
        backend_type='userpass',
        mount_point=userpass_auth_path,
    )

client.mfa.configure(
    mount_point=userpass_auth_path,
)
```

Reading the MFA Auth Method Configuration

hvac.api.auth.Mfa.read_configuration()

```
import hvac
client = hvac.Client()

mfa_configuration = client.mfa.read_configuration()
print('The MFA auth method is configured with a MFA type of: {mfa_type}'.format(
    mfa_type=mfa_configuration['data']['type']
)
```

Configure Duo MFA Type Access Credentials

hvac.api.auth.Mfa.configure_duo_access()

```
from getpass import getpass

import hvac
client = hvac.Client()

secret_key_prompt = 'Please enter the Duo access secret key to configure: '
duo_access_secret_key = getpass(prompt=secret_key_prompt)

client.mfa.configure_duo_access(
    mount_point=userpass_auth_path,
    host='api-1234abcd.duosecurity.com',
    integration_key='SOME_DUO_IKEY',
    secret_key=duo_access_secret_key,
)
```

Configure Duo MFA Type Behavior

hvac.api.auth.Mfa.configure_duo_behavior()

```
import hvac
client = hvac.Client()

client.mfa.configure_duo_behavior(
    mount_point=userpass_auth_path,
    username_format='%s@hvac.network',
)
```

Read Duo MFA Type Behavior

hvac.api.auth.Mfa.read_duo_behavior_configuration()

```
import hvac
client = hvac.Client()

duo_behavior_config = client.mfa.read_duo_behavior_configuration(
    mount_point=userpass_auth_path,
)
print('The Duo MFA behvaior is configured with a username_format of: {username_format}
↪'.format(
    username_format=duo_behavior_config['data']['username_format'],
)
```

Authentication / Login

```
from getpass import getpass

import hvac

login_username = 'someuser'
password_prompt = 'Please enter your password for the userpass (with MFA)␣
↪authentication backend: '
login_password = getpass(prompt=password_prompt)
passcode_prompt = 'Please enter your OTP for the userpass (with MFA) authentication␣
↪backend: '
userpass_mfa_passcode = getpass(prompt=passcode_prompt)

client = hvac.Client()

# Here the mount_point parameter corresponds to the path provided when enabling the␣
↪backend
client.mfa.auth_userpass(
    username=login_username,
    password=login_password,
    mount_point=userpass_auth_path,
    passcode=userpass_mfa_passcode,
)
print(client.is_authenticated)  # => True
```

2.2.8 Token

Authentication

```
# Token
client.token = 'MY_TOKEN'
assert client.is_authenticated() # => True
```

Token Management

Token creation and revocation:

```
token = client.create_token(policies=['root'], lease='1h')

current_token = client.lookup_token()
some_other_token = client.lookup_token('xxx')

client.revoke_token('xxx')
client.revoke_token('yyy', orphan=True)

client.revoke_token_prefix('zzz')

client.renew_token('aaa')
```

Lookup and revoke tokens via a token accessor:

```
token = client.create_token(policies=['root'], lease='1h')
token_accessor = token['auth']['accessor']

same_token = client.lookup_token(token_accessor, accessor=True)
client.revoke_token(token_accessor, accessor=True)
```

Wrapping/unwrapping a token:

```
wrap = client.create_token(policies=['root'], lease='1h', wrap_ttl='1m')
result = self.client.unwrap(wrap['wrap_info']['token'])
```

2.2.9 Authenticate to different auth backends

```
# App ID
client.auth_app_id('MY_APP_ID', 'MY_USER_ID')

# GitHub
client.auth_github('MY_GITHUB_TOKEN')

# TLS
client = Client(cert=('path/to/cert.pem', 'path/to/key.pem'))
client.auth_tls()

# Non-default mount point (available on all auth types)
client.auth_userpass('MY_USERNAME', 'MY_PASSWORD', mount_point='CUSTOM_MOUNT_POINT')

# Authenticating without changing to new token (available on all auth types)
```

(continues on next page)

(continued from previous page)

```
result = client.auth_github('MY_GITHUB_TOKEN', use_token=False)
print(result['auth']['client_token']) # => u'NEW_TOKEN'

# Custom or unsupported auth type
params = {
    'username': 'MY_USERNAME',
    'password': 'MY_PASSWORD',
    'custom_param': 'MY_CUSTOM_PARAM',
}

result = client.auth('/v1/auth/CUSTOM_AUTH/login', json=params)

# Logout
client.logout()
```

2.3 System Backend

2.3.1 Initialize and seal/unseal

```
print(client.is_initialized()) # => False

shares = 5
threshold = 3

result = client.initialize(shares, threshold)

root_token = result['root_token']
keys = result['keys']

print(client.is_initialized()) # => True

print(client.is_sealed()) # => True

# unseal with individual keys
client.unseal(keys[0])
client.unseal(keys[1])
client.unseal(keys[2])

# unseal with multiple keys until threshold met
client.unseal_multi(keys)

print(client.is_sealed()) # => False

client.seal()

print(client.is_sealed()) # => True
```

2.3.2 Manipulate auth backends

```
backends = client.list_auth_backends()
```

(continues on next page)

(continued from previous page)

```
client.enable_auth_backend('userpass', mount_point='customuserpass')
client.disable_auth_backend('github')
```

2.3.3 Manipulate secret backends

```
backends = client.list_secret_backends()

client.enable_secret_backend('aws', mount_point='aws-us-east-1')
client.disable_secret_backend('mysql')

client.tune_secret_backend('generic', mount_point='test', default_lease_ttl='3600s',␣
↪max_lease_ttl='8600s')
client.get_secret_backend_tuning('generic', mount_point='test')

client.remount_secret_backend('aws-us-east-1', 'aws-east')
```

2.3.4 Manipulate policies

```
policies = client.list_policies() # => ['root']

policy = """
path "sys" {
  policy = "deny"
}

path "secret" {
  policy = "write"
}

path "secret/foo" {
  policy = "read"
}
"""

client.set_policy('myapp', policy)

client.delete_policy('oldthing')

policy = client.get_policy('mypolicy')

# Requires pyhcl to automatically parse HCL into a Python dictionary
policy = client.get_policy('mypolicy', parse=True)
```

2.3.5 Manipulate audit backends

```
backends = client.list_audit_backends()

options = {
    'path': '/tmp/vault.log',
    'log_raw': True,
```

(continues on next page)

(continued from previous page)

```
}

client.enable_audit_backend('file', options=options, name='somefile')
client.disable_audit_backend('oldfile')
```

2.3.6 View and Manage Leases

Read a lease:

New in version 0.6.2.

```
>>> client.read_lease(lease_id='pki/issue/my-role/d05138a2-edeb-889d-db98-2057ecd5138f
↪')
{'lease_id': '', 'warnings': None, 'wrap_info': None, 'auth': None, 'lease_duration':␣
↪0, 'request_id': 'a08768dc-b14e-5e2d-f291-4702056f8d4e', 'data': {'last_renewal':␣
↪None, 'ttl': 259145, 'expire_time': '2018-07-19T06:20:02.000046424-05:00', 'id':
↪'pki/issue/my-role/d05138a2-edeb-889d-db98-2057ecd5138f', 'renewable': False,
↪'issue_time': '2018-07-16T06:20:02.918474523-05:00'}, 'renewable': False}
```

Renewing a lease:

```
>>> client.renew_secret(lease_id='pki/issue/my-role/d05138a2-edeb-889d-db98-
↪2057ecd5138f')
{'lease_id': 'pki/issue/my-role/d05138a2-edeb-889d-db98-2057ecd5138f', 'lease_duration
↪': 2764790, 'renewable': True}
```

Revoking a lease:

```
>>> client.revoke_secret(lease_id='pki/issue/my-role/d05138a2-edeb-889d-db98-
↪2057ecd5138f')
```

CHAPTER 3

Advanced Usage

3.1 Making Use of Private CA

There is a not uncommon use case of people deploying Hashicorp Vault with a private certificate authority. Unfortunately the *requests* module does not make use of the system CA certificates. Instead of disabling SSL verification you can make use of the *REQUESTS_CA_BUNDLE* environment variable.

As documented in the advanced usage section for requests this environment variable should point to a file that is comprised of all CA certificates you may wish to use. This can be a single private CA, or an existing list of root certificates with the private appended to the end. The following example shows how to achieve this:

```
$ cp "$(python -c 'import certifi;print certifi.where();')" /tmp/bundle.pem
$ cat /path/to/custom.pem >> /tmp/bundle.pem
$ export REQUESTS_CA_BUNDLE=/tmp/bundle.pem
```

Alternative, this envrionmental variable can be set via the *os* module in-line with other Python statements. The following example would be one way to manage this configuration on a Ubuntu host:

```
import os

import hvac


def get_vault_client(vault_url=VAULT_URL, certs=VAULT_CERTS):
        """
        Instantiates a hvac / vault client.
        :param vault_url: string, protocol + address + port for the vault service
        :param certs: tuple, Optional tuple of self-signed certs to use for␣
↪verification
                with hvac's requests adapater.
        :return: hvac.Client
        """
        logger.debug('Retrieving a vault (hvac) client...')
        if certs:
```

(continues on next page)

(continued from previous page)

```
                # When use a self-signed certificate for the vault service itself, we␣
↪need to
                # include our local ca bundle here for the underlying requests module.
                os.environ['REQUESTS_CA_BUNDLE'] = '/etc/ssl/certs/ca-certificates.crt
↪'

        vault_client = hvac.Client(
                url=vault_url,
                cert=certs,
        )

        vault_client.token = load_vault_token(vault_client)

        if not vault_client.is_authenticated():
                error_msg = 'Unable to authenticate to the Vault service'
                raise hvac.exceptions.Unauthorized(error_msg)

        return vault_client
```

3.2 Custom Requests / HTTP Adapter

New in version 0.6.2.

Calls to the requests module. (which provides the methods hvac utilizes to send HTTP/HTTPS request to Vault instances) were extracted from the *Client* class and moved to a newly added *hvac.adapters()* module. The *Client* class itself defaults to an instance of the *Request* class for its _adapter private attribute attribute if no adapter argument is provided to its *constructor*. This attribute provides an avenue for modifying the manner in which hvac completes request. To enable this type of customization, implement a class of type *hvac.adapters.Adapter()*, override its abstract methods, and pass an instance of this custom class to the adapter argument of the *Client constructor*

CHAPTER 4

Source Reference

4.1 hvac.v1

class hvac.v1.**Client**(*url=u'http://localhost:8200'*, *token=None*, *cert=None*, *verify=True*, *timeout=30*, *proxies=None*, *allow_redirects=True*, *session=None*, *adapter=None*)

Bases: object

The hvac Client class for HashiCorp's Vault.

__init__(*url=u'http://localhost:8200'*, *token=None*, *cert=None*, *verify=True*, *timeout=30*, *proxies=None*, *allow_redirects=True*, *session=None*, *adapter=None*)

Creates a new hvac client instnace.

Parameters

- **url** (*str*) – Base URL for the Vault instance being addressed.
- **token** (*str*) – Authentication token to include in requests sent to Vault.
- **cert** (*tuple*) – Certificates for use in requests sent to the Vault instance. This should be a tuple with the certificate and then key.
- **verify** (*Union[bool,str]*) – Either a boolean to indicate whether TLS verification should be performed when sending requests to Vault, or a string pointing at the CA bundle to use for verification. See http://docs.python-requests.org/en/master/user/advanced/#ssl-cert-verification.
- **timeout** (*int*) – The timeout value for requests sent to Vault.
- **proxies** (*dict*) – Proxies to use when preforming requests. See: http://docs.python-requests.org/en/master/user/advanced/#proxies
- **allow_redirects** (*bool*) – Whether to follow redirects when sending requests to Vault.
- **session** (*request.Session*) – Optional session object to use when performing request.

- **adapter** (hvac.adapters.Adapter) – Optional class to be used for performing requests. If none is provided, defaults to hvac.adapters.Request

adapter

allow_redirects

audit_hash(*name*, *input*)
POST /sys/audit-hash

Parameters

- **name** –
- **input** –

Returns

Return type

auth(*url*, *use_token=True*, ***kwargs*)

Performs a request (typically to a path prefixed with "/v1/auth") and optionaly stores the client token sent in the resulting Vault response for use by the *hvac.adapters.Adapter()* instance under the _adapater Client attribute.

Parameters

- **url** (*str | unicode*) – Path to send the authentication request to.
- **use_token** (*bool*) – if True, uses the token in the response received from the auth request to set the "token" attribute on the the *hvac.adapters.Adapter()* instance under the _adapater Client attribute.
- **kwargs** (*dict*) – Additional keyword arguments to include in the params sent with the request.

Returns The response of the auth request.

Return type requests.Response

auth_app_id(*app_id*, *user_id*, *mount_point=u'app-id'*, *use_token=True*)
POST /auth/<mount point>/login

Parameters

- **app_id** –
- **user_id** –
- **mount_point** –
- **use_token** –

Returns

Return type

auth_approle(*role_id*, *secret_id=None*, *mount_point=u'approle'*, *use_token=True*)
POST /auth/<mount_point>/login

Parameters

- **role_id** –
- **secret_id** –

- **mount_point** –
- **use_token** –

Returns

Return type

auth_aws_iam(*access_key*, *secret_key*, *session_token=None*, *header_value=None*, *mount_point=u'aws'*, *role=u''*, *use_token=True*, *region=u'us-east-1'*)
POST /auth/<mount point>/login

Parameters

- **access_key** (*str*) – AWS IAM access key ID
- **secret_key** (*str*) – AWS IAM secret access key
- **session_token** (*str*) – Optional AWS IAM session token retrieved via a GetSessionToken AWS API request. see: https://docs.aws.amazon.com/STS/latest/APIReference/API_GetSessionToken.html
- **header_value** (*str*) – Vault allows you to require an additional header, X-Vault-AWS-IAM-Server-ID, to be present to mitigate against different types of replay attacks. Depending on the configuration of the AWS auth backend, providing a argument to this optional parameter may be required.
- **mount_point** (*str*) – The "path" the AWS auth backend was mounted on. Vault currently defaults to "aws". "aws-ec2" is the default argument for backwards comparability within this module.
- **role** (*str*) – Name of the role against which the login is being attempted. If role is not specified, then the login endpoint looks for a role bearing the name of the AMI ID of the EC2 instance that is trying to login if using the ec2 auth method, or the "friendly name" (i.e., role name or username) of the IAM principal authenticated. If a matching role is not found, login fails.
- **use_token** (*bool.*) – If True, uses the token in the response received from the auth request to set the "token" attribute on the current Client class instance.

Returns The response from the AWS IAM login request attempt.

Return type requests.Response

auth_cubbyhole(*token*)
POST /v1/sys/wrapping/unwrap

Parameters **token** –

Returns

Return type

auth_ec2(*pkcs7*, *nonce=None*, *role=None*, *use_token=True*, *mount_point=u'aws-ec2'*)
POST /auth/<mount point>/login

Parameters

- **pkcs7** (*str.*) – PKCS#7 version of an AWS Instance Identity Document from the EC2 Metadata Service.
- **nonce** (*str.*) – Optional nonce returned as part of the original authentication request. Not required if the backend has "allow_instance_migration" or "disallow_reauthentication" options turned on.
- **role** (*str.*) – Identifier for the AWS auth backend role being requested.

- **use_token** (*bool.*) – If True, uses the token in the response received from the auth request to set the "token" attribute on the current Client class instance.
- **mount_point** (*str.*) – The "path" the AWS auth backend was mounted on. Vault currently defaults to "aws". "aws-ec2" is the default argument for backwards comparability within this module.

Returns parsed JSON response from the auth POST request

Return type dict.

auth_gcp(*role*, *jwt*, *mount_point=u'gcp'*, *use_token=True*)
POST /auth/<mount point>/login

Parameters

- **role** (*str.*) – identifier for the GCP auth backend role being requested
- **jwt** (*str.*) – JSON Web Token from the GCP metadata service
- **mount_point** (*str.*) – The "path" the GCP auth backend was mounted on. Vault currently defaults to "gcp".
- **use_token** (*bool.*) – if True, uses the token in the response received from the auth request to set the "token" attribute on the current Client class instance.

Returns parsed JSON response from the auth POST request

Return type dict.

auth_github(***kwargs*)

Call to deprecated function 'auth_github'. This method will be removed in version '0.8.0' Please use the 'login' meth
Docstring content from this method's replacement copied below: Login using GitHub access token.

Supported methods: POST: /auth/{mount_point}/login. Produces: 200 application/json

Parameters

- **token** (*str | unicode*) – GitHub personal API token.
- **use_token** (*bool*) – if True, uses the token in the response received from the auth request to set the "token" attribute on the the *hvac.adapters.Adapter()* instance under the _adapater Client attribute.
- **mount_point** (*str | unicode*) – The "path" the method/backend was mounted on.

Returns The JSON response of the login request.

Return type dict

auth_kubernetes(*role*, *jwt*, *use_token=True*, *mount_point=u'kubernetes'*)
POST /auth/<mount_point>/login

Parameters

- **role** (*str.*) – Name of the role against which the login is being attempted.
- **jwt** (*str.*) – Signed JSON Web Token (JWT) for authenticating a service account.
- **use_token** (*bool.*) – if True, uses the token in the response received from the auth request to set the "token" attribute on the current Client class instance.
- **mount_point** (*str.*) – The "path" the k8s auth backend was mounted on. Vault currently defaults to "kubernetes".

Returns Parsed JSON response from the config POST request.

Return type dict.

auth_ldap(***kwargs*)

Call to deprecated function 'auth_ldap'. This method will be removed in version '0.8.0' Please use the 'login' method
Docstring content from this method's replacement copied below:

Log in with LDAP credentials.

Supported methods: POST: /auth/{mount_point}/login/{username}. Produces: 200 application/json

Parameters

- **username** (*str | unicode*) – The username of the LDAP user
- **password** (*str | unicode*) – The password for the LDAP user
- **use_token** (*bool*) – if True, uses the token in the response received from the auth request to set the "token" attribute on the the *hvac.adapters.Adapter()* instance under the _adapater Client attribute.
- **mount_point** (*str | unicode*) – The "path" the method/backend was mounted on.

Returns The response of the login_with_user request.

Return type requests.Response

auth_tls(*mount_point=u'cert'*, *use_token=True*)
POST /auth/<mount point>/login

Parameters

- **mount_point** –
- **use_token** –

Returns

Return type

auth_userpass(*username*, *password*, *mount_point=u'userpass'*, *use_token=True*, ***kwargs*)
POST /auth/<mount point>/login/<username>

Parameters

- **username** –
- **password** –
- **mount_point** –
- **use_token** –
- **kwargs** –

Returns

Return type

cancel_generate_root()
DELETE /sys/generate-root/attempt

Returns

Return type

cancel_rekey()
DELETE /sys/rekey/init

Returns

Return type

close(***kwargs*)
Call to deprecated function 'close'. This method will be removed in version '0.8.0' Please use the 'close' method on the 'hvac.adapters' class moving forward. Docstring content from this method's replacement copied below: Close the underlying Requests session.

create_app_id(*app_id*, *policies*, *display_name=None*, *mount_point=u'app-id'*, ***kwargs*)
POST /auth/<mount point>/map/app-id/<app_id>

Parameters

- **app_id** –
- **policies** –
- **display_name** –
- **mount_point** –
- **kwargs** –

Returns

Return type

create_ec2_role(*role*, *bound_ami_id=None*, *bound_account_id=None*, *bound_iam_role_arn=None*, *bound_iam_instance_profile_arn=None*, *bound_ec2_instance_id=None*, *bound_region=None*, *bound_vpc_id=None*, *bound_subnet_id=None*, *role_tag=None*, *ttl=None*, *max_ttl=None*, *period=None*, *policies=None*, *allow_instance_migration=False*, *disallow_reauthentication=False*, *resolve_aws_unique_ids=None*, *mount_point=u'aws-ec2'*)
POST /auth/<mount_point>/role/<role>

Parameters

- **role** –
- **bound_ami_id** –
- **bound_account_id** –
- **bound_iam_role_arn** –
- **bound_iam_instance_profile_arn** –
- **bound_ec2_instance_id** –
- **bound_region** –
- **bound_vpc_id** –
- **bound_subnet_id** –
- **role_tag** –
- **ttl** –
- **max_ttl** –
- **period** –

- **policies** –
- **allow_instance_migration** –
- **disallow_reauthentication** –
- **resolve_aws_unique_ids** –
- **mount_point** –

Returns

Return type

create_ec2_role_tag(*role*, *policies=None*, *max_ttl=None*, *instance_id=None*, *disallow_reauthentication=False*, *allow_instance_migration=False*, *mount_point=u'aws-ec2'*)

POST /auth/<mount_point>/role/<role>/tag

Parameters

- **role** –
- **policies** –
- **max_ttl** –
- **instance_id** –
- **disallow_reauthentication** –
- **allow_instance_migration** –
- **mount_point** –

Returns

Return type

create_kubernetes_configuration(*kubernetes_host*, *kubernetes_ca_cert=None*, *token_reviewer_jwt=None*, *pem_keys=None*, *mount_point=u'kubernetes'*)

POST /auth/<mount_point>/config

Parameters

- **kubernetes_host** (*str.*) – A host:port pair, or a URL to the base of the Kubernetes API server.
- **kubernetes_ca_cert** (*str.*) – PEM encoded CA cert for use by the TLS client used to talk with the Kubernetes API.
- **token_reviewer_jwt** (*str.*) – A service account JWT used to access the TokenReview API to validate other JWTs during login. If not set the JWT used for login will be used to access the API.
- **pem_keys** (*list.*) – 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.
- **mount_point** (*str.*) – The "path" the k8s auth backend was mounted on. Vault currently defaults to "kubernetes".

Returns Will be an empty body with a 204 status code upon success

Return type requests.Response.

create_kubernetes_role(*name*, *bound_service_account_names*, *bound_service_account_namespaces*, *ttl=u''*, *max_ttl=u''*, *period=u''*, *policies=None*, *mount_point=u'kubernetes'*)

POST /auth/<mount_point>/role/:name

Parameters

- **name** (*str.*) – Name of the role.
- **bound_service_account_names** (*list.*) – List of service account names able to access this role. If set to "*" all names are allowed, both this and bound_service_account_namespaces can not be "*".
- **bound_service_account_namespaces** (*list.*) – List of namespaces allowed to access this role. If set to "*" all namespaces are allowed, both this and bound_service_account_names can not be set to "*".
- **ttl** (*str.*) – The TTL period of tokens issued using this role in seconds.
- **max_ttl** (*str.*) – The maximum allowed lifetime of tokens issued in seconds using this role.
- **period** (*str.*) – If set, indicates that the token generated using this role should never expire. The token should be renewed within the duration specified by this value. At each renewal, the token's TTL will be set to the value of this parameter.
- **policies** (*list.*) – Policies to be set on tokens issued using this role
- **mount_point** (*str.*) – The "path" the k8s auth backend was mounted on. Vault currently defaults to "kubernetes".

Returns Will be an empty body with a 204 status code upon success

Return type requests.Response.

create_role(*role_name*, *mount_point=u'approle'*, ***kwargs*)

POST /auth/<mount_point>/role/<role name>

Parameters

- **role_name** –
- **mount_point** –
- **kwargs** –

Returns

Return type

create_role_custom_secret_id(*role_name*, *secret_id*, *meta=None*, *mount_point=u'approle'*)

POST /auth/<mount_point>/role/<role name>/custom-secret-id

Parameters

- **role_name** –
- **secret_id** –
- **meta** –
- **mount_point** –

Returns

Return type

create_role_secret_id(*role_name*, *meta=None*, *cidr_list=None*, *wrap_ttl=None*, *mount_point=u'approle'*)

POST /auth/<mount_point>/role/<role name>/secret-id

Parameters

- **role_name** –
- **meta** –
- **cidr_list** –
- **wrap_ttl** –
- **mount_point** –

Returns

Return type

create_token(*role=None*, *token_id=None*, *policies=None*, *meta=None*, *no_parent=False*, *lease=None*, *display_name=None*, *num_uses=None*, *no_default_policy=False*, *ttl=None*, *orphan=False*, *wrap_ttl=None*, *renewable=None*, *explicit_max_ttl=None*, *period=None*)

POST /auth/token/create

POST /auth/token/create/<role>

POST /auth/token/create-orphan

Parameters

- **role** –
- **token_id** –
- **policies** –
- **meta** –
- **no_parent** –
- **lease** –
- **display_name** –
- **num_uses** –
- **no_default_policy** –
- **ttl** –
- **orphan** –
- **wrap_ttl** –
- **renewable** –
- **explicit_max_ttl** –
- **period** –

Returns

Return type

create_token_role(*role*, *allowed_policies=None*, *disallowed_policies=None*, *orphan=None*, *period=None*, *renewable=None*, *path_suffix=None*, *explicit_max_ttl=None*)

POST /auth/token/roles/<role>

Parameters

- **role** –
- **allowed_policies** –
- **disallowed_policies** –
- **orphan** –
- **period** –
- **renewable** –
- **path_suffix** –
- **explicit_max_ttl** –

Returns

Return type

create_user_id(*user_id*, *app_id*, *cidr_block=None*, *mount_point=u'app-id'*, ***kwargs*)

POST /auth/<mount point>/map/user-id/<user_id>

Parameters

- **user_id** –
- **app_id** –
- **cidr_block** –
- **mount_point** –
- **kwargs** –

Returns

Return type

create_userpass(*username*, *password*, *policies*, *mount_point=u'userpass'*, ***kwargs*)

POST /auth/<mount point>/users/<username>

Parameters

- **username** –
- **password** –
- **policies** –
- **mount_point** –
- **kwargs** –

Returns

Return type

create_vault_ec2_certificate_configuration(*cert_name*, *aws_public_cert*, *mount_point=u'aws-ec2'*)

POST /auth/<mount_point>/config/certificate/<cert_name>

Parameters

- **cert_name** –
- **aws_public_cert** –
- **mount_point** –

Returns

Return type

create_vault_ec2_client_configuration(*access_key*, *secret_key*, *endpoint=None*, *mount_point=u'aws-ec2'*)

POST /auth/<mount_point>/config/client

Parameters

- **access_key** –
- **secret_key** –
- **endpoint** –
- **mount_point** –

Returns

Return type

delete(*path*)

DELETE /<path>

Parameters **path** –

Returns

Return type

delete_app_id(*app_id*, *mount_point=u'app-id'*)

DELETE /auth/<mount_point>/map/app-id/<app_id>

Parameters

- **app_id** –
- **mount_point** –

Returns

Return type

delete_ec2_role(*role*, *mount_point=u'aws-ec2'*)

DELETE /auth/<mount_point>/role/<role>

Parameters

- **role** –
- **mount_point** –

Returns

Return type

delete_kubernetes_role(*role*, *mount_point=u'kubernetes'*)

DELETE /auth/<mount_point>/role/:role

Parameters

- **role** (*Name of the role.*) – str.
- **mount_point** (*str.*) – The "path" the k8s auth backend was mounted on. Vault currently defaults to "kubernetes".

Returns Will be an empty body with a 204 status code upon success.

Return type requests.Response.

delete_policy(*name*)
DELETE /sys/policy/<name>

Parameters **name** –

Returns

Return type

delete_role(*role_name*, *mount_point=u'approle'*)
DELETE /auth/<mount_point>/role/<role name>

Parameters

- **role_name** –
- **mount_point** –

Returns

Return type

delete_role_secret_id(*role_name*, *secret_id*, *mount_point=u'approle'*)
POST /auth/<mount_point>/role/<role name>/secret-id/destroy

Parameters

- **role_name** –
- **secret_id** –
- **mount_point** –

Returns

Return type

delete_role_secret_id_accessor(*role_name*, *secret_id_accessor*, *mount_point=u'approle'*)
DELETE /auth/<mount_point>/role/<role name>/secret-id/<secret_id_accessor>

Parameters

- **role_name** –
- **secret_id_accessor** –
- **mount_point** –

Returns

Return type

delete_token_role(*role*)
Deletes the named token role.

Parameters **role** –

Returns

Return type

delete_user_id(*user_id*, *mount_point=u'app-id'*)
DELETE /auth/<mount_point>/map/user-id/<user_id>

Parameters

- **user_id** –
- **mount_point** –

Returns

Return type

delete_userpass(*username*, *mount_point=u'userpass'*)
DELETE /auth/<mount point>/users/<username>

Parameters

- **username** –
- **mount_point** –

Returns

Return type

delete_vault_ec2_client_configuration(*mount_point=u'aws-ec2'*)
DELETE /auth/<mount_point>/config/client

Parameters **mount_point** –

Returns

Return type

disable_audit_backend(*name*)
DELETE /sys/audit/<name>

Parameters **name** –

Returns

Return type

disable_auth_backend(*mount_point*)
DELETE /sys/auth/<mount point>

Parameters **mount_point** –

Returns

Return type

disable_secret_backend(*mount_point*)
DELETE /sys/mounts/<mount point>

Parameters **mount_point** –

Returns

Return type

enable_audit_backend(*backend_type*, *description=None*, *options=None*, *name=None*)
POST /sys/audit/<name>

Parameters

- **backend_type** –
- **description** –
- **options** –
- **name** –

Returns

Return type

enable_auth_backend(*backend_type*, *description=None*, *mount_point=None*, *config=None*, *plugin_name=None*)
POST /sys/auth/<mount point>

Parameters

- **backend_type** –
- **description** –
- **mount_point** –
- **config** –
- **plugin_name** –

Returns

Return type

enable_secret_backend(*backend_type*, *description=None*, *mount_point=None*, *config=None*, *options=None*)
POST /sys/mounts/<mount point>

Parameters

- **backend_type** –
- **description** –
- **mount_point** –
- **config** –
- **options** –

Returns

Return type

generate_root(*key*, *nonce*)
PUT /sys/generate-root/update

Parameters

- **key** –
- **nonce** –

Returns

Return type

generate_root_status
GET /sys/generate-root/attempt

Returns

Return type

get_app_id(*app_id*, *mount_point=u'app-id'*, *wrap_ttl=None*)
GET /auth/<mount_point>/map/app-id/<app_id>

Parameters

- **app_id** –
- **mount_point** –
- **wrap_ttl** –

Returns

Return type

get_auth_backend_tuning(*backend_type*, *mount_point=None*)
GET /sys/auth/<mount point>/tune

Parameters

- **backend_type** (*str.*) – Name of the auth backend to modify (e.g., token, approle, etc.)
- **mount_point** (*str.*) – The path the associated auth backend is mounted under.

Returns The JSON response from Vault

Return type dict.

get_backed_up_keys()
GET /sys/rekey/backup

Returns

Return type

get_ec2_role(*role*, *mount_point=u'aws-ec2'*)
GET /auth/<mount_point>/role/<role>

Parameters

- **role** –
- **mount_point** –

Returns

Return type

get_kubernetes_configuration(*mount_point=u'kubernetes'*)
GET /auth/<mount_point>/config

Parameters **mount_point** (*str.*) – The "path" the k8s auth backend was mounted on. Vault currently defaults to "kubernetes".

Returns Parsed JSON response from the config GET request

Return type dict.

get_kubernetes_role(*name*, *mount_point=u'kubernetes'*)
GET /auth/<mount_point>/role/:name

Parameters

- **name** (*str.*) – Name of the role.
- **mount_point** (*str.*) – The "path" the k8s auth backend was mounted on. Vault currently defaults to "kubernetes".

Returns Parsed JSON response from the read role GET request

Return type dict.

get_policy(*name*, *parse=False*)
GET /sys/policy/<name>

Parameters

- **name** –

- **parse** –

Returns

Return type

get_role(*role_name*, *mount_point=u'approle'*)
GET /auth/<mount_point>/role/<role name>

Parameters

- **role_name** –
- **mount_point** –

Returns

Return type

get_role_id(*role_name*, *mount_point=u'approle'*)
GET /auth/<mount_point>/role/<role name>/role-id

Parameters

- **role_name** –
- **mount_point** –

Returns

Return type

get_role_secret_id(*role_name*, *secret_id*, *mount_point=u'approle'*)
POST /auth/<mount_point>/role/<role name>/secret-id/lookup

Parameters

- **role_name** –
- **secret_id** –
- **mount_point** –

Returns

Return type

get_role_secret_id_accessor(*role_name*, *secret_id_accessor*, *mount_point=u'approle'*)
POST /auth/<mount_point>/role/<role name>/secret-id-accessor/lookup

Parameters

- **role_name** –
- **secret_id_accessor** –
- **mount_point** –

Returns

Return type

get_secret_backend_tuning(*backend_type*, *mount_point=None*)
GET /sys/mounts/<mount point>/tune

Parameters

- **backend_type** –
- **mount_point** –

Returns

Return type

get_user_id(*user_id*, *mount_point=u'app-id'*, *wrap_ttl=None*)
GET /auth/<mount_point>/map/user-id/<user_id>

Parameters

- **user_id** –
- **mount_point** –
- **wrap_ttl** –

Returns

Return type

get_vault_ec2_certificate_configuration(*cert_name*, *mount_point=u'aws-ec2'*)
GET /auth/<mount_point>/config/certificate/<cert_name>

Parameters

- **cert_name** –
- **mount_point** –

Returns

Return type

get_vault_ec2_client_configuration(*mount_point=u'aws-ec2'*)
GET /auth/<mount_point>/config/client

Parameters **mount_point** –

Returns

Return type

github
Accessor for the Client instance's Github methods. Provided via the *hvac.api.auth.Github* class.

Returns This Client instance's associated Github instance.

Return type *hvac.api.auth.Github*

ha_status
GET /sys/leader

Returns

Return type

initialize(*secret_shares=5*, *secret_threshold=3*, *pgp_keys=None*)
PUT /sys/init

Parameters

- **secret_shares** –
- **secret_threshold** –
- **pgp_keys** –

Returns

Return type

is_authenticated()
Helper method which returns the authentication status of the client

Returns

Return type

is_initialized()
GET /sys/init

Returns

Return type

is_sealed()

Returns

Return type

key_status
GET /sys/key-status

Returns

Return type

kv
Accessor for the Client instance's KV methods. Provided via the *hvac.api.secrets_engines.Kv* class.

Returns This Client instance's associated Kv instance.

Return type *hvac.api.secrets_engines.Kv*

ldap
Accessor for the Client instance's LDAP methods. Provided via the *hvac.api.auth.Ldap* class.

Returns This Client instance's associated Ldap instance.

Return type *hvac.api.auth.Ldap*

list(*path*)
GET /<path>?list=true

Parameters **path** –

Returns

Return type

list_audit_backends()
GET /sys/audit

Returns

Return type

list_auth_backends()
GET /sys/auth

Returns

Return type

list_ec2_roles(*mount_point=u'aws-ec2'*)
GET /auth/<mount_point>/roles?list=true

Parameters **mount_point** –

Returns

Return type

list_kubernetes_roles(*mount_point=u'kubernetes'*)
GET /auth/<mount_point>/role?list=true

Parameters **mount_point** (*str.*) – The "path" the k8s auth backend was mounted on. Vault currently defaults to "kubernetes".

Returns Parsed JSON response from the list roles GET request.

Return type dict.

list_policies()
GET /sys/policy

Returns

Return type

list_role_secrets(*role_name*, *mount_point=u'approle'*)
GET /auth/<mount_point>/role/<role name>/secret-id?list=true

Parameters

- **role_name** –
- **mount_point** –

Returns

Return type

list_roles(*mount_point=u'approle'*)
GET /auth/<mount_point>/role

Parameters **mount_point** –

Returns

Return type

list_secret_backends()
GET /sys/mounts

Returns

Return type

list_token_roles()
GET /auth/token/roles?list=true

Returns

Return type

list_userpass(*mount_point=u'userpass'*)
GET /auth/<mount point>/users?list=true

Parameters **mount_point** –

Returns

Return type

list_vault_ec2_certificate_configurations(*mount_point=u'aws-ec2'*)
GET /auth/<mount_point>/config/certificates?list=true

Parameters **mount_point** –

Returns

Return type

logout(*revoke_token=False*)
Clears the token used for authentication, optionally revoking it before doing so.

Parameters **revoke_token** –

Returns

Return type

lookup_token(*token=None*, *accessor=False*, *wrap_ttl=None*)
GET /auth/token/lookup/<token>

GET /auth/token/lookup-accessor/<token-accessor>

GET /auth/token/lookup-self

Parameters

- **token** (*str.*) –
- **accessor** (*str.*) –
- **wrap_ttl** (*int.*) –

Returns

Return type

mfa
Accessor for the Client instance's MFA methods. Provided via the `hvac.api.auth.mfa` class.

Returns This Client instance's associated MFA instance.

Return type hvac.api.auth.mfa

read(*path*, *wrap_ttl=None*)
GET /<path>

Parameters

- **path** –
- **wrap_ttl** –

Returns

Return type

read_lease(*lease_id*)
PUT /sys/leases/lookup

Parameters **lease_id** (*str.*) – Specifies the ID of the lease to lookup.

Returns Parsed JSON response from the leases PUT request

Return type dict.

read_userpass(*username*, *mount_point=u'userpass'*)
GET /auth/<mount point>/users/<username>

Parameters

- **username** –
- **mount_point** –

Returns

Return type

rekey(*key*, *nonce=None*)
PUT /sys/rekey/update

Parameters

- **key** –
- **nonce** –

Returns

Return type

rekey_multi(*keys*, *nonce=None*)

Parameters

- **keys** –
- **nonce** –

Returns

Return type

rekey_status
GET /sys/rekey/init

Returns

Return type

remount_secret_backend(*from_mount_point*, *to_mount_point*)
POST /sys/remount

Parameters

- **from_mount_point** –
- **to_mount_point** –

Returns

Return type

renew_secret(*lease_id*, *increment=None*)
PUT /sys/leases/renew

Parameters

- **lease_id** –
- **increment** –

Returns

Return type

renew_token(*token=None*, *increment=None*, *wrap_ttl=None*)
POST /auth/token/renew

POST /auth/token/renew-self

Parameters

- **token** –
- **increment** –
- **wrap_ttl** –

Returns

Return type

revoke_secret(*lease_id*)
PUT /sys/revoke/<lease id>

Parameters **lease_id** –

Returns

Return type

revoke_secret_prefix(*path_prefix*)
PUT /sys/revoke-prefix/<path prefix>

Parameters **path_prefix** –

Returns

Return type

revoke_self_token()
PUT /auth/token/revoke-self

Returns

Return type

revoke_token(*token*, *orphan=False*, *accessor=False*)
POST /auth/token/revoke

POST /auth/token/revoke-orphan

POST /auth/token/revoke-accessor

Parameters

- **token** –
- **orphan** –
- **accessor** –

Returns

Return type

revoke_token_prefix(*prefix*)
POST /auth/token/revoke-prefix/<prefix>

Parameters **prefix** –

Returns

Return type

rotate()
PUT /sys/rotate

Returns

Return type

seal()
PUT /sys/seal

Returns

Return type

seal_status
GET /sys/seal-status

Returns

Return type

session

set_policy(*name*, *rules*)
PUT /sys/policy/<name>

Parameters

- **name** –
- **rules** –

Returns

Return type

set_role_id(*role_name*, *role_id*, *mount_point=u'approle'*)
POST /auth/<mount_point>/role/<role name>/role-id

Parameters

- **role_name** –
- **role_id** –
- **mount_point** –

Returns

Return type

start_generate_root(*key*, *otp=False*)
PUT /sys/generate-root/attempt

Parameters

- **key** –
- **otp** –

Returns

Return type

start_rekey(*secret_shares=5*, *secret_threshold=3*, *pgp_keys=None*, *backup=False*)
PUT /sys/rekey/init

Parameters

- **secret_shares** –
- **secret_threshold** –
- **pgp_keys** –
- **backup** –

Returns

Return type

token

token_role(*role*)
Returns the named token role.

Parameters **role** –

Returns

Return type

transit_create_key(*name*, *convergent_encryption=None*, *derived=None*, *exportable=None*, *key_type=None*, *mount_point=u'transit'*)
POST /<mount_point>/keys/<name>

Parameters

- **name** –
- **convergent_encryption** –
- **derived** –
- **exportable** –
- **key_type** –
- **mount_point** –

Returns

Return type

transit_decrypt_data(*name*, *ciphertext*, *context=None*, *nonce=None*, *batch_input=None*, *mount_point=u'transit'*)
POST /<mount_point>/decrypt/<name>

Parameters

- **name** –
- **ciphertext** –
- **context** –
- **nonce** –
- **batch_input** –
- **mount_point** –

Returns

Return type

transit_delete_key(*name*, *mount_point=u'transit'*)
DELETE /<mount_point>/keys/<name>

Parameters

- **name** –
- **mount_point** –

Returns

Return type

transit_encrypt_data(*name*, *plaintext*, *context=None*, *key_version=None*, *nonce=None*, *batch_input=None*, *key_type=None*, *convergent_encryption=None*, *mount_point=u'transit'*)

POST /<mount_point>/encrypt/<name>

Parameters

- **name** –
- **plaintext** –
- **context** –
- **key_version** –
- **nonce** –
- **batch_input** –
- **key_type** –
- **convergent_encryption** –
- **mount_point** –

Returns

Return type

transit_export_key(*name*, *key_type*, *version=None*, *mount_point=u'transit'*)

GET /<mount_point>/export/<key_type>/<name>(/<version>)

Parameters

- **name** –
- **key_type** –
- **version** –
- **mount_point** –

Returns

Return type

transit_generate_data_key(*name*, *key_type*, *context=None*, *nonce=None*, *bits=None*, *mount_point=u'transit'*)

POST /<mount_point>/datakey/<type>/<name>

Parameters

- **name** –
- **key_type** –
- **context** –
- **nonce** –

- **bits** –
- **mount_point** –

Returns

Return type

transit_generate_hmac(*name*, *hmac_input*, *key_version=None*, *algorithm=None*, *mount_point=u'transit'*)

POST /<mount_point>/hmac/<name>(/<algorithm>)

Parameters

- **name** –
- **hmac_input** –
- **key_version** –
- **algorithm** –
- **mount_point** –

Returns

Return type

transit_generate_rand_bytes(*data_bytes=None*, *output_format=None*, *mount_point=u'transit'*)

POST /<mount_point>/random(/<data_bytes>)

Parameters

- **data_bytes** –
- **output_format** –
- **mount_point** –

Returns

Return type

transit_hash_data(*hash_input*, *algorithm=None*, *output_format=None*, *mount_point=u'transit'*)

POST /<mount_point>/hash(/<algorithm>)

Parameters

- **hash_input** –
- **algorithm** –
- **output_format** –
- **mount_point** –

Returns

Return type

transit_list_keys(*mount_point=u'transit'*)

GET /<mount_point>/keys?list=true

Parameters **mount_point** –

Returns

Return type

transit_read_key(*name*, *mount_point=u'transit'*)
GET /<mount_point>/keys/<name>

Parameters

- **name** –
- **mount_point** –

Returns

Return type

transit_rewrap_data(*name*, *ciphertext*, *context=None*, *key_version=None*, *nonce=None*, *batch_input=None*, *mount_point=u'transit'*)
POST /<mount_point>/rewrap/<name>

Parameters

- **name** –
- **ciphertext** –
- **context** –
- **key_version** –
- **nonce** –
- **batch_input** –
- **mount_point** –

Returns

Return type

transit_rotate_key(*name*, *mount_point=u'transit'*)
POST /<mount_point>/keys/<name>/rotate

Parameters

- **name** –
- **mount_point** –

Returns

Return type

transit_sign_data(*name*, *input_data*, *key_version=None*, *algorithm=None*, *context=None*, *prehashed=None*, *mount_point=u'transit'*, *signature_algorithm=u'pss'*)
POST /<mount_point>/sign/<name>(/<algorithm>)

Parameters

- **name** –
- **input_data** –
- **key_version** –
- **algorithm** –
- **context** –
- **prehashed** –
- **mount_point** –

- **signature_algorithm** –

Returns

Return type

transit_update_key(*name*, *min_decryption_version=None*, *min_encryption_version=None*, *deletion_allowed=None*, *mount_point=u'transit'*)

POST /<mount_point>/keys/<name>/config

Parameters

- **name** –
- **min_decryption_version** –
- **min_encryption_version** –
- **deletion_allowed** –
- **mount_point** –

Returns

Return type

transit_verify_signed_data(*name*, *input_data*, *algorithm=None*, *signature=None*, *hmac=None*, *context=None*, *prehashed=None*, *mount_point=u'transit'*, *signature_algorithm=u'pss'*)

POST /<mount_point>/verify/<name>(/<algorithm>)

Parameters

- **name** –
- **input_data** –
- **algorithm** –
- **signature** –
- **hmac** –
- **context** –
- **prehashed** –
- **mount_point** –
- **signature_algorithm** –

Returns

Return type

tune_auth_backend(*backend_type*, *mount_point=None*, *default_lease_ttl=None*, *max_lease_ttl=None*, *description=None*, *audit_non_hmac_request_keys=None*, *audit_non_hmac_response_keys=None*, *listing_visibility=None*, *passthrough_request_headers=None*)

POST /sys/auth/<mount point>/tune

Parameters

- **backend_type** (*str.*) – Name of the auth backend to modify (e.g., token, approle, etc.)
- **mount_point** (*str.*) – The path the associated auth backend is mounted under.

- **description** (*str.*) – Specifies the description of the mount. This overrides the current stored value, if any.
- **default_lease_ttl** (*int.*) –
- **max_lease_ttl** (*int.*) –
- **audit_non_hmac_request_keys** (*list.*) – Specifies the comma-separated list of keys that will not be HMAC'd by audit devices in the request data object.
- **audit_non_hmac_response_keys** (*list.*) – Specifies the comma-separated list of keys that will not be HMAC'd by audit devices in the response data object.
- **listing_visibility** (*str.*) – Specifies whether to show this mount in the UI-specific listing endpoint. Valid values are "unauth" or "".
- **passthrough_request_headers** (*list.*) – Comma-separated list of headers to whitelist and pass from the request to the backend.

Returns The JSON response from Vault

Return type dict.

tune_secret_backend(*backend_type*, *mount_point=None*, *default_lease_ttl=None*, *max_lease_ttl=None*, *description=None*, *audit_non_hmac_request_keys=None*, *audit_non_hmac_response_keys=None*, *listing_visibility=None*, *passthrough_request_headers=None*)

POST /sys/mounts/<mount point>/tune

Parameters

- **backend_type** (*str*) – Type of the secret backend to modify
- **mount_point** (*str*) – The path the associated secret backend is mounted
- **description** (*str*) – Specifies the description of the mount. This overrides the current stored value, if any.
- **default_lease_ttl** (*int*) – Default time-to-live. This overrides the global default. A value of 0 is equivalent to the system default TTL
- **max_lease_ttl** (*int*) – Maximum time-to-live. This overrides the global default. A value of 0 are equivalent and set to the system max TTL.
- **audit_non_hmac_request_keys** (*list*) – Specifies the comma-separated list of keys that will not be HMAC'd by audit devices in the request data object.
- **audit_non_hmac_response_keys** (*list*) – Specifies the comma-separated list of keys that will not be HMAC'd by audit devices in the response data object.
- **listing_visibility** (*str*) – Speficies whether to show this mount in the UI-specific listing endpoint. Valid values are "unauth" or "".
- **passthrough_request_headers** (*str*) – Comma-separated list of headers to whitelist and pass from the request to the backend.

Returns The JSON response from Vault

Return type dict.

unseal(*key*)

PUT /sys/unseal

Parameters **key** –

Returns

Return type

unseal_multi(*keys*)

Parameters **keys** –

Returns

Return type

unseal_reset()
PUT /sys/unseal

Returns

Return type

unwrap(*token=None*)
POST /sys/wrapping/unwrap

Parameters **token** –

Returns

Return type

update_userpass_password(*username*, *password*, *mount_point=u'userpass'*)
POST /auth/<mount point>/users/<username>/password

Parameters

- **username** –
- **password** –
- **mount_point** –

Returns

Return type

update_userpass_policies(*username*, *policies*, *mount_point=u'userpass'*)
POST /auth/<mount point>/users/<username>/policies

Parameters

- **username** –
- **policies** –
- **mount_point** –

Returns

Return type

url

static urljoin(**args*, ***kwargs*)

Call to deprecated function 'urljoin'. This method will be removed in version '0.8.0' Please use the 'urljoin' method
Docstring content from this method's replacement copied below: Joins given arguments into a url. Trailing and leading slashes are stripped for each argument.

Parameters **args** (*str | unicode*) – Multiple parts of a URL to be combined into one string.

Returns Full URL combining all provided arguments

Return type str | unicode

write(*path*, *wrap_ttl=None*, ***kwargs*)
POST /<path>

Parameters

- **path** –
- **wrap_ttl** –
- **kwargs** –

Returns

Return type

4.2 hvac.api

Collection of Vault API endpoint classes.

class hvac.api.**VaultApiBase**(*adapter*)
Bases: object

Base class for API endpoints.

__init__(*adapter*)
Default api class constructor.

Parameters **adapter** (hvac.adapters.Adapter) – Instance of *hvac.adapters.Adapter*; used for performing HTTP requests.

4.3 hvac.api.auth

Collection of classes for various Vault auth methods.

class hvac.api.auth.**Github**(*adapter*)
Bases: hvac.api.vault_api_base.VaultApiBase

GitHub Auth Method (API).

Reference: https://www.vaultproject.io/api/auth/github/index.html

configure(*organization*, *base_url=''*, *ttl=''*, *max_ttl=''*, *mount_point='github'*)
Configure the connection parameters for GitHub.

This path honors the distinction between the create and update capabilities inside ACL policies.

Supported methods: POST: /auth/{mount_point}/config. Produces: 204 (empty body)

Parameters

- **organization** (*str | unicode*) – The organization users must be part of.
- **base_url** (*str | unicode*) – The API endpoint to use. Useful if you are running GitHub Enterprise or an API-compatible authentication server.
- **ttl** (*str | unicode*) – Duration after which authentication will be expired.
- **max_ttl** (*str | unicode*) – Maximum duration after which authentication will be expired.

- **mount_point** (*str | unicode*) – The "path" the method/backend was mounted on.

Returns The response of the configure_method request.

Return type requests.Response

login(*token*, *use_token=True*, *mount_point='github'*)
Login using GitHub access token.

Supported methods: POST: /auth/{mount_point}/login. Produces: 200 application/json

Parameters

- **token** (*str | unicode*) – GitHub personal API token.
- **use_token** (*bool*) – if True, uses the token in the response received from the auth request to set the "token" attribute on the the *hvac.adapters.Adapter()* instance under the _adapater Client attribute.
- **mount_point** (*str | unicode*) – The "path" the method/backend was mounted on.

Returns The JSON response of the login request.

Return type dict

map_team(*team_name*, *policies=None*, *mount_point='github'*)
Map a list of policies to a team that exists in the configured GitHub organization.

Supported methods: POST: /auth/{mount_point}/map/teams/{team_name}. Produces: 204 (empty body)

Parameters

- **team_name** (*str | unicode*) – GitHub team name in "slugified" format
- **policies** (*list*) – Comma separated list of policies to assign
- **mount_point** (*str | unicode*) – The "path" the method/backend was mounted on.

Returns The response of the map_github_teams request.

Return type requests.Response

map_user(*user_name*, *policies=None*, *mount_point='github'*)
Map a list of policies to a specific GitHub user exists in the configured organization.

Supported methods: POST: /auth/{mount_point}/map/users/{user_name}. Produces: 204 (empty body)

Parameters

- **user_name** (*str | unicode*) – GitHub user name
- **policies** (*str | unicode*) – Comma separated list of policies to assign
- **mount_point** (*str | unicode*) – The "path" the method/backend was mounted on.

Returns The response of the map_github_users request.

Return type requests.Response

read_configuration(*mount_point='github'*)
Read the GitHub configuration.

Supported methods: GET: /auth/{mount_point}/config. Produces: 200 application/json

Parameters **mount_point** (*str | unicode*) – The "path" the method/backend was mounted on.

Returns The JSON response of the read_configuration request.

Return type dict

read_team_mapping(*team_name*, *mount_point='github'*)

Read the GitHub team policy mapping.

Supported methods: GET: /auth/{mount_point}/map/teams/{team_name}. Produces: 200 application/json

Parameters

- **team_name** (*str | unicode*) – GitHub team name
- **mount_point** (*str | unicode*) – The "path" the method/backend was mounted on.

Returns The JSON response of the read_team_mapping request.

Return type dict

read_user_mapping(*user_name*, *mount_point='github'*)

Read the GitHub user policy mapping.

Supported methods: GET: /auth/{mount_point}/map/users/{user_name}. Produces: 200 application/json

Parameters

- **user_name** (*str | unicode*) – GitHub user name
- **mount_point** (*str | unicode*) – The "path" the method/backend was mounted on.

Returns The JSON response of the read_user_mapping request.

Return type dict

class hvac.api.auth.**Ldap**(*adapter*)

Bases: hvac.api.vault_api_base.VaultApiBase

LDAP Auth Method (API).

Reference: https://www.vaultproject.io/api/auth/ldap/index.html

configure(*user_dn*, *group_dn*, *url='ldap://127.0.0.1'*, *case_sensitive_names=False*, *starttls=False*, *tls_min_version='tls12'*, *tls_max_version='tls12'*, *insecure_tls=False*, *certificate=None*, *bind_dn=None*, *bind_pass=None*, *user_attr='cn'*, *discover_dn=False*, *deny_null_bind=True*, *upn_domain=None*, *group_filter='(|(memberUid={{.Username}})(member={{.UserDN}})(uniqueMember={{.UserDN}}))'*, *group_attr='cn'*, *mount_point='ldap'*)

Configure the LDAP auth method.

Supported methods: POST: /auth/{mount_point}/config. Produces: 204 (empty body)

Parameters

- **user_dn** (*str | unicode*) – Base DN under which to perform user search. Example: ou=Users,dc=example,dc=com

- **group_dn** (*str | unicode*) – 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
- **url** (*str | unicode*) – The LDAP server to connect to. Examples: ldap://ldap.myorg.com, ldaps://ldap.myorg.com:636. Multiple URLs can be specified with commas, e.g. ldap://ldap.myorg.com,ldap://ldap2.myorg.com; these will be tried in-order.
- **case_sensitive_names** (*bool*) – If set, user and group names assigned to policies within the backend will be case sensitive. Otherwise, names will be normalized to lower case. Case will still be preserved when sending the username to the LDAP server at login time; this is only for matching local user/group definitions.
- **starttls** (*bool*) – If true, issues a StartTLS command after establishing an unencrypted connection.
- **tls_min_version** (*str | unicode*) – Minimum TLS version to use. Accepted values are tls10, tls11 or tls12.
- **tls_max_version** (*str | unicode*) – Maximum TLS version to use. Accepted values are tls10, tls11 or tls12.
- **insecure_tls** (*bool*) – If true, skips LDAP server SSL certificate verification - insecure, use with caution!
- **certificate** (*str | unicode*) – CA certificate to use when verifying LDAP server certificate, must be x509 PEM encoded.
- **bind_dn** (*str | unicode*) – Distinguished name of object to bind when performing user search. Example: cn=vault,ou=Users,dc=example,dc=com
- **bind_pass** (*str | unicode*) – Password to use along with binddn when performing user search.
- **user_attr** (*str | unicode*) – Attribute on user attribute object matching the username passed when authenticating. Examples: sAMAccountName, cn, uid
- **discover_dn** (*bool*) – Use anonymous bind to discover the bind DN of a user.
- **deny_null_bind** (*bool*) – This option prevents users from bypassing authentication when providing an empty password.
- **upn_domain** (*str | unicode*) – The 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 vault to bind as username@example.com.
- **group_filter** (*str | unicode*) – 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}})).
- **group_attr** (*str | unicode*) – 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.
- **mount_point** (*str | unicode*) – The "path" the method/backend was mounted on.

Returns The response of the configure request.

Return type requests.Response

create_or_update_group(*name*, *policies=None*, *mount_point='ldap'*)
Create or update LDAP group policies.

Supported methods: POST: /auth/{mount_point}/groups/{name}. Produces: 204 (empty body)

Parameters

- **name** (*str | unicode*) – The name of the LDAP group
- **policies** (*list*) – List of policies associated with the group. This parameter is transformed to a comma-delimited string before being passed to Vault.
- **mount_point** (*str | unicode*) – The "path" the method/backend was mounted on.

Returns The response of the create_or_update_group request.

Return type requests.Response

create_or_update_user(*username*, *policies=None*, *groups=None*, *mount_point='ldap'*)
Create or update LDAP users policies and group associations.

Supported methods: POST: /auth/{mount_point}/users/{username}. Produces: 204 (empty body)

Parameters

- **username** (*str | unicode*) – The username of the LDAP user
- **policies** (*str | unicode*) – List of policies associated with the user. This parameter is transformed to a comma-delimited string before being passed to Vault.
- **groups** (*str | unicode*) – List of groups associated with the user. This parameter is transformed to a comma-delimited string before being passed to Vault.
- **mount_point** (*str | unicode*) – The "path" the method/backend was mounted on.

Returns The response of the create_or_update_user request.

Return type requests.Response

delete_group(*name*, *mount_point='ldap'*)
Delete a LDAP group and policy association.

Supported methods: DELETE: /auth/{mount_point}/groups/{name}. Produces: 204 (empty body)

Parameters

- **name** (*str | unicode*) – The name of the LDAP group
- **mount_point** (*str | unicode*) – The "path" the method/backend was mounted on.

Returns The response of the delete_group request.

Return type requests.Response

delete_user(*username*, *mount_point='ldap'*)
Delete a LDAP user and policy association.

Supported methods: DELETE: /auth/{mount_point}/users/{username}. Produces: 204 (empty body)

Parameters

- **username** (*str | unicode*) – The username of the LDAP user
- **mount_point** (*str | unicode*) – The "path" the method/backend was mounted on.

Returns The response of the delete_user request.

Return type requests.Response

list_groups(*mount_point='ldap'*)
List existing LDAP existing groups that have been created in this auth method.

Supported methods: LIST: /auth/{mount_point}/groups. Produces: 200 application/json

Parameters **mount_point** (*str | unicode*) – The "path" the method/backend was mounted on.

Returns The JSON response of the list_groups request.

Return type dict

list_users(*mount_point='ldap'*)
List existing users in the method.

Supported methods: LIST: /auth/{mount_point}/users. Produces: 200 application/json

Parameters **mount_point** (*str | unicode*) – The "path" the method/backend was mounted on.

Returns The JSON response of the list_users request.

Return type dict

login(*username*, *password*, *use_token=True*, *mount_point='ldap'*)
Log in with LDAP credentials.

Supported methods: POST: /auth/{mount_point}/login/{username}. Produces: 200 application/json

Parameters

- **username** (*str | unicode*) – The username of the LDAP user
- **password** (*str | unicode*) – The password for the LDAP user
- **use_token** (*bool*) – if True, uses the token in the response received from the auth request to set the "token" attribute on the the *hvac.adapters.Adapter()* instance under the _adapater Client attribute.
- **mount_point** (*str | unicode*) – The "path" the method/backend was mounted on.

Returns The response of the login_with_user request.

Return type requests.Response

read_configuration(*mount_point='ldap'*)
Retrieve the LDAP configuration for the auth method.

Supported methods: GET: /auth/{mount_point}/config. Produces: 200 application/json

Parameters **mount_point** (*str | unicode*) – The "path" the method/backend was mounted on.

Returns The JSON response of the read_configuration request.

Return type dict

read_group(*name*, *mount_point='ldap'*)
Read policies associated with a LDAP group.

Supported methods: GET: /auth/{mount_point}/groups/{name}. Produces: 200 application/json

Parameters

- **name** (*str | unicode*) – The name of the LDAP group
- **mount_point** (*str | unicode*) – The "path" the method/backend was mounted on.

Returns The JSON response of the read_group request.

Return type dict

read_user(*username*, *mount_point='ldap'*)
Read policies associated with a LDAP user.

Supported methods: GET: /auth/{mount_point}/users/{username}. Produces: 200 application/json

Parameters

- **username** (*str | unicode*) – The username of the LDAP user
- **mount_point** (*str | unicode*) – The "path" the method/backend was mounted on.

Returns The JSON response of the read_user request.

Return type dict

class hvac.api.auth.**Mfa**(*adapter*)
Bases: hvac.api.vault_api_base.VaultApiBase

Multi-factor authentication Auth Method (API).

Warning: This class's methods correspond to a legacy / unsupported set of Vault API routes. Please see the reference link for additional context.

Reference: https://www.vaultproject.io/docs/auth/mfa.html

configure(*mount_point*, *mfa_type='duo'*, *force=False*)
Configure MFA for a supported method.

This endpoint allows you to turn on multi-factor authentication with a given backend. Currently only Duo is supported.

Supported methods: POST: /auth/{mount_point}/mfa_config. Produces: 204 (empty body)

Parameters

- **mount_point** (*str | unicode*) – The "path" the method/backend was mounted on.
- **mfa_type** (*str | unicode*) – Enables MFA with given backend (available: duo)
- **force** (*bool*) – If True, make the "mfa_config" request regardless of circumstance. If False (the default), verify the provided mount_point is available and one of the types of methods supported by this feature.

Returns The response of the configure MFA request.

Return type requests.Response

configure_duo_access(*mount_point*, *host*, *integration_key*, *secret_key*)
Configure the access keys and host for Duo API connections.

To authenticate users with Duo, the backend needs to know what host to connect to and must authenticate with an integration key and secret key. This endpoint is used to configure that information.

Supported methods: POST: /auth/{mount_point}/duo/access. Produces: 204 (empty body)

Parameters

- **mount_point** (*str | unicode*) – The "path" the method/backend was mounted on.
- **host** (*str | unicode*) – Duo API host
- **integration_key** (*Duo secret key*) – Duo integration key
- **secret_key** (*str | unicode*) – The "path" the method/backend was mounted on.

Returns The response of the configure_duo_access request.

Return type requests.Response

configure_duo_behavior(*mount_point*, *push_info=None*, *user_agent=None*, *username_format='%s'*)
Configure Duo second factor behavior.

This endpoint allows you to configure how the original auth method username maps to the Duo username by providing a template format string.

Supported methods: POST: /auth/{mount_point}/duo/config. Produces: 204 (empty body)

Parameters

- **mount_point** (*str | unicode*) – The "path" the method/backend was mounted on.
- **push_info** (*str | unicode*) – A string of URL-encoded key/value pairs that provides additional context about the authentication attempt in the Duo Mobile app
- **user_agent** (*str | unicode*) – User agent to connect to Duo (default "")
- **username_format** (*str | unicode*) – Format string given auth method username as argument to create Duo username (default '%s')

Returns The response of the configure_duo_behavior request.

Return type requests.Response

read_configuration(*mount_point*)
Read the MFA configuration.

Supported methods: GET: /auth/{mount_point}/mfa_config. Produces: 200 application/json

Parameters **mount_point** (*str | unicode*) – The "path" the method/backend was mounted on.

Returns The JSON response of the read_configuration request.

Return type dict

read_duo_behavior_configuration(*mount_point*)
Read the Duo second factor behavior configuration.

Supported methods: GET: /auth/{mount_point}/duo/config. Produces: 200 application/json

Parameters **mount_point** (*str | unicode*) – The "path" the method/backend was mounted on.

Returns The JSON response of the read_duo_behavior_configuration request.

Return type dict

4.4 hvac.api.secrets_engines

Vault secrets engines endpoints

class hvac.api.secrets_engines.**Kv**(*adapter*, *default_kv_version='2'*)

Bases: hvac.api.vault_api_base.VaultApiBase

Class containing methods for the key/value secrets_engines backend API routes. Reference: https://www.vaultproject.io/docs/secrets/kv/index.html

__init__(*adapter*, *default_kv_version='2'*)

Create a new Kv instnace.

Parameters

- **adapter** (hvac.adapters.Adapter) – Instance of *hvac.adapters.Adapter*; used for performing HTTP requests.
- **default_kv_version** (*str | unicode*) – KV version number (e.g., '1') to use as the default when accessing attributes/methods under this class.

allowed_kv_versions = ['1', '2']

default_kv_version

v1

Accessor for kv version 1 class / method. Provided via the hvac.api.secrets_engines.kv_v1.KvV1 class.

Returns This Kv instance's associated KvV1 instance.

Return type hvac.api.secrets_engines.kv_v1.KvV1

v2

Accessor for kv version 2 class / method. Provided via the hvac.api.secrets_engines.kv_v2.KvV2 class.

Returns This Kv instance's associated KvV2 instance.

Return type hvac.api.secrets_engines.kv_v2.KvV2

class hvac.api.secrets_engines.**KvV1**(*adapter*)

Bases: hvac.api.vault_api_base.VaultApiBase

KV Secrets Engine - Version 1 (API).

Reference: https://www.vaultproject.io/api/secrets/kv/kv-v1.html

create_or_update_secret(*path*, *secret*, *method=None*, *mount_point='secret'*)

Store a secret at the specified location.

If the value does not yet exist, the calling token must have an ACL policy granting the create capability. If the value already exists, the calling token must have an ACL policy granting the update capability.

Supported methods: POST: /{mount_point}/{path}. Produces: 204 (empty body) PUT: /{mount_point}/{path}. Produces: 204 (empty body)

Parameters

- **path** (*str | unicode*) – Specifies the path of the secrets to create/update. This is specified as part of the URL.
- **secret** (*dict*) – Specifies keys, paired with associated values, to be held at the given location. Multiple key/value pairs can be specified, and all will be returned on a read operation. A key called ttl will trigger some special behavior. See the Vault KV secrets engine documentation for details.
- **method** (*str | unicode*) – Optional parameter to explicitly request a POST (create) or PUT (update) request to the selected kv secret engine. If no argument is provided for this parameter, hvac attempts to intelligently determine which method is appropriate.
- **mount_point** (*str | unicode*) – The "path" the secret engine was mounted on.

Returns The response of the create_or_update_secret request.

Return type requests.Response

delete_secret(*path*, *mount_point='secret'*)

Delete the secret at the specified location.

Supported methods: DELETE: /{mount_point}/{path}. Produces: 204 (empty body)

Parameters

- **path** (*str | unicode*) – Specifies the path of the secret to delete. This is specified as part of the URL.
- **mount_point** (*str | unicode*) – The "path" the secret engine was mounted on.

Returns The response of the delete_secret request.

Return type requests.Response

list_secrets(*path*, *mount_point='secret'*)

Return a list of key names at the specified location.

Folders are suffixed with /. The input must be a folder; list on a file will not return a value. Note that no policy-based filtering is performed on keys; do not encode sensitive information in key names. The values themselves are not accessible via this command.

Supported methods: LIST: /{mount_point}/{path}. Produces: 200 application/json

Parameters

- **path** (*str | unicode*) – Specifies the path of the secrets to list. This is specified as part of the URL.
- **mount_point** (*str | unicode*) – The "path" the secret engine was mounted on.

Returns The JSON response of the list_secrets request.

Return type dict

read_secret(*path*, *mount_point='secret'*)

Retrieve the secret at the specified location.

Supported methods: GET: /{mount_point}/{path}. Produces: 200 application/json

Parameters

- **path** (*str | unicode*) – Specifies the path of the secret to read. This is specified as part of the URL.
- **mount_point** (*str | unicode*) – The "path" the secret engine was mounted on.

Returns The JSON response of the read_secret request.

Return type dict

class hvac.api.secrets_engines.**KvV2**(*adapter*)

Bases: hvac.api.vault_api_base.VaultApiBase

KV Secrets Engine - Version 2 (API).

Reference: https://www.vaultproject.io/api/secret/kv/kv-v2.html

configure(*max_versions=10*, *cas_required=None*, *mount_point='secret'*)

Configure backend level settings that are applied to every key in the key-value store.

Supported methods: POST: /{mount_point}/config. Produces: 204 (empty body)

Parameters

- **max_versions** (*int*) – The number of versions to keep per key. This value applies to all keys, but a key's metadata setting can overwrite this value. Once a key has more than the configured allowed versions the oldest version will be permanently deleted. Defaults to 10.
- **cas_required** (*bool*) – If true all keys will require the cas parameter to be set on all write requests.
- **mount_point** (*str | unicode*) – The "path" the secret engine was mounted on.

Returns The response of the request.

Return type requests.Response

create_or_update_secret(*path*, *secret*, *cas=None*, *mount_point='secret'*)

Create a new version of a secret at the specified location.

If the value does not yet exist, the calling token must have an ACL policy granting the create capability. If the value already exists, the calling token must have an ACL policy granting the update capability.

Supported methods: POST: /{mount_point}/data/{path}. Produces: 200 application/json

Parameters

- **path** (*str | unicode*) – Path
- **cas** (*int*) – 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.
- **secret** (*dict*) – The contents of the "secret" dict will be stored and returned on read.
- **mount_point** (*str | unicode*) – The "path" the secret engine was mounted on.

Returns The JSON response of the request.

Return type dict

delete_latest_version_of_secret(*path*, *mount_point='secret'*)
Issue a soft delete of the secret's latest version at the specified location.

This marks the version as deleted and will stop it from being returned from reads, but the underlying data will not be removed. A delete can be undone using the undelete path.

Supported methods: DELETE: /{mount_point}/data/{path}. Produces: 204 (empty body)

Parameters

- **path** (*str | unicode*) – Specifies the path of the secret to delete. This is specified as part of the URL.
- **mount_point** (*str | unicode*) – The "path" the secret engine was mounted on.

Returns The response of the request.

Return type requests.Response

delete_metadata_and_all_versions(*path*, *mount_point='secret'*)
Delete (permanently) the key metadata and all version data for the specified key.

All version history will be removed.

Supported methods: DELETE: /{mount_point}/metadata/{path}. Produces: 204 (empty body)

Parameters

- **path** (*str | unicode*) – Specifies the path of the secret to delete. This is specified as part of the URL.
- **mount_point** (*str | unicode*) – The "path" the secret engine was mounted on.

Returns The response of the request.

Return type requests.Response

delete_secret_versions(*path*, *versions*, *mount_point='secret'*)
Issue a soft delete of the specified versions of the secret.

This marks the versions as deleted and will stop them from being returned from reads, but the underlying data will not be removed. A delete can be undone using the undelete path.

Supported methods: POST: /{mount_point}/delete/{path}. Produces: 204 (empty body)

Parameters

- **path** (*str | unicode*) – Specifies the path of the secret to delete. This is specified as part of the URL.
- **versions** (*int*) – The versions to be deleted. The versioned data will not be deleted, but it will no longer be returned in normal get requests.
- **mount_point** (*str | unicode*) – The "path" the secret engine was mounted on.

Returns The response of the request.

Return type requests.Response

destroy_secret_versions(*path*, *versions*, *mount_point='secret'*)
Permanently remove the specified version data and numbers for the provided path from the key-value store.

Supported methods: POST: /{mount_point}/destroy/{path}. Produces: 204 (empty body)

Parameters

- **path** (*str | unicode*) – Specifies the path of the secret to destroy. This is specified as part of the URL.
- **versions** (*list of int*) – The versions to destroy. Their data will be permanently deleted.
- **mount_point** (*str | unicode*) – The "path" the secret engine was mounted on.

Returns The response of the request.

Return type requests.Response

list_secrets(*path*, *mount_point='secret'*)

Return a list of key names at the specified location.

Folders are suffixed with /. The input must be a folder; list on a file will not return a value. Note that no policy-based filtering is performed on keys; do not encode sensitive information in key names. The values themselves are not accessible via this command.

Supported methods: LIST: /{mount_point}/metadata/{path}. Produces: 200 application/json

Parameters

- **path** (*str | unicode*) – Specifies the path of the secrets to list. This is specified as part of the URL.
- **mount_point** (*str | unicode*) – The "path" the secret engine was mounted on.

Returns The JSON response of the request.

Return type dict

read_configuration(*mount_point='secret'*)

Read the KV Version 2 configuration.

Supported methods: GET: /auth/{mount_point}/config. Produces: 200 application/json

Parameters **mount_point** (*str | unicode*) – The "path" the secret engine was mounted on.

Returns The JSON response of the request.

Return type dict

read_secret_metadata(*path*, *mount_point='secret'*)

Retrieve the metadata and versions for the secret at the specified path.

Supported methods: GET: /{mount_point}/metadata/{path}. Produces: 200 application/json

Parameters

- **path** (*str | unicode*) – Specifies the path of the secret to read. This is specified as part of the URL.
- **mount_point** (*str | unicode*) – The "path" the secret engine was mounted on.

Returns The JSON response of the request.

Return type dict

read_secret_version(*path*, *version=None*, *mount_point='secret'*)
Retrieve the secret at the specified location.

Supported methods: GET: /{mount_point}/data/{path}. Produces: 200 application/json

Parameters

- **path** (*str | unicode*) – Specifies the path of the secret to read. This is specified as part of the URL.
- **version** (*int*) – Specifies the version to return. If not set the latest version is returned.
- **mount_point** (*str | unicode*) – The "path" the secret engine was mounted on.

Returns The JSON response of the request.

Return type dict

undelete_secret_versions(*path*, *versions*, *mount_point='secret'*)
Undelete the data for the provided version and path in the key-value store.

This restores the data, allowing it to be returned on get requests.

Supported methods: POST: /{mount_point}/undelete/{path}. Produces: 204 (empty body)

Parameters

- **path** (*str | unicode*) – Specifies the path of the secret to undelete. This is specified as part of the URL.
- **versions** (*list of int*) – The versions to undelete. The versions will be restored and their data will be returned on normal get requests.
- **mount_point** (*str | unicode*) – The "path" the secret engine was mounted on.

Returns The response of the request.

Return type requests.Response

update_metadata(*path*, *max_versions=None*, *cas_required=None*, *mount_point='secret'*)
Updates the max_versions of cas_required setting on an existing path.

Supported methods: POST: /{mount_point}/metadata/{path}. Produces: 204 (empty body)

Parameters

- **path** (*str | unicode*) – Path
- **max_versions** (*int*) – The number of versions to keep per key. If not set, the backend's configured max version is used. Once a key has more than the configured allowed versions the oldest version will be permanently deleted.
- **cas_required** (*bool*) – 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.
- **mount_point** (*str | unicode*) – The "path" the secret engine was mounted on.

Returns The response of the request.

Return type requests.Response

4.5 hvac.utils

Misc utility functions and constants

hvac.utils.**deprecated_method**(*to_be_removed_in_version*, *new_method=None*)

This is a decorator which can be used to mark methods as deprecated. It will result in a warning being emitted when the function is used.

Parameters

- **to_be_removed_in_version** (*str*) – Version of this module the decorated method will be removed in.
- **new_method** (*function*) – Method intended to replace the decorated method. This method's docstrings are included in the decorated method's docstring.

Returns Wrapped function that includes a deprecation warning and update docstrings from the replacement method.

Return type types.FunctionType

hvac.utils.**raise_for_error**(*status_code*, *message=None*, *errors=None*)

Helper method to raise exceptions based on the status code of a response received back from Vault.

Parameters

- **status_code** (*int*) – Status code received in a response from Vault.
- **message** (*str*) – Optional message to include in a resulting exception.
- **errors** (*list | str*) – Optional errors to include in a resulting exception.

Raises hvac.exceptions.InvalidRequest | hvac.exceptions.Unauthorized | hvac.exceptions.Forbidden | hvac.exceptions.InvalidPath | hvac.exceptions.RateLimitExceeded | hvac.exceptions.InternalServerError | hvac.exceptions.VaultNotInitialized | hvac.exceptions.VaultDown | hvac.exceptions.UnexpectedError

4.6 hvac.aws_utils

class hvac.aws_utils.**SigV4Auth**(*access_key*, *secret_key*, *session_token=None*, *region='us-east-1'*)

Bases: object

__init__(*access_key*, *secret_key*, *session_token=None*, *region='us-east-1'*)

x.__init__(...) initializes x; see help(type(x)) for signature

add_auth(*request*)

hvac.aws_utils.**generate_sigv4_auth_request**(*header_value=None*)

Helper function to prepare a AWS API request to subsequently generate a "AWS Signature Version 4" header.

Parameters **header_value** (*str*) – Vault allows you to require an additional header, X-Vault-AWS-IAM-Server-ID, to be present to mitigate against different types of replay attacks. Depending on the configuration of the AWS auth backend, providing a argument to this optional parameter may be required.

Returns A PreparedRequest instance, optionally containing the provided header value under a 'X-Vault-AWS-IAM-Server-ID' header name pointed to AWS's simple token service with action "GetCallerIdentity"

Return type requests.PreparedRequest

4.7 hvac.adapters

HTTP Client Library Adapters

class hvac.adapters.**Adapter**(*base_uri='http://localhost:8200'*, *token=None*, *cert=None*, *verify=True*, *timeout=30*, *proxies=None*, *allow_redirects=True*, *session=None*)

Bases: object

Abstract base class used when constructing adapters for use with the Client class.

__init__(*base_uri='http://localhost:8200'*, *token=None*, *cert=None*, *verify=True*, *timeout=30*, *proxies=None*, *allow_redirects=True*, *session=None*)

Create a new request adapter instance.

Parameters

- **base_uri** (*str*) – Base URL for the Vault instance being addressed.
- **token** (*str*) – Authentication token to include in requests sent to Vault.
- **cert** (*tuple*) – Certificates for use in requests sent to the Vault instance. This should be a tuple with the certificate and then key.
- **verify** (*Union[bool,str]*) – Either a boolean to indicate whether TLS verification should be performed when sending requests to Vault, or a string pointing at the CA bundle to use for verification. See http://docs.python-requests.org/en/master/user/advanced/#ssl-cert-verification.
- **timeout** (*int*) – The timeout value for requests sent to Vault.
- **proxies** (*dict*) – Proxies to use when preforming requests. See: http://docs.python-requests.org/en/master/user/advanced/#proxies
- **allow_redirects** (*bool*) – Whether to follow redirects when sending requests to Vault.
- **session** (*request.Session*) – Optional session object to use when performing request.

auth(*url*, *use_token=True*, ***kwargs*)

Performs a request (typically to a path prefixed with "/v1/auth") and optionaly stores the client token sent in the resulting Vault response for use by the *hvac.adapters.Adapter()* instance under the _adapater Client attribute.

Parameters

- **url** (*str | unicode*) – Path to send the authentication request to.
- **use_token** (*bool*) – if True, uses the token in the response received from the auth request to set the "token" attribute on the the *hvac.adapters.Adapter()* instance under the _adapater Client attribute.
- **kwargs** (*dict*) – Additional keyword arguments to include in the params sent with the request.

Returns The response of the auth request.

Return type requests.Response

close()

Close the underlying Requests session.

delete(*url*, ***kwargs*)
Performs a DELETE request.

Parameters

- **url** (*str | unicode*) – Partial URL path to send the request to. This will be joined to the end of the instance's base_uri attribute.
- **kwargs** (*dict*) – Additional keyword arguments to include in the requests call.

Returns The response of the request.

Return type requests.Response

get(*url*, ***kwargs*)
Performs a GET request.

Parameters

- **url** (*str | unicode*) – Partial URL path to send the request to. This will be joined to the end of the instance's base_uri attribute.
- **kwargs** (*dict*) – Additional keyword arguments to include in the requests call.

Returns The response of the request.

Return type requests.Response

list(*url*, ***kwargs*)
Performs a LIST request.

Parameters

- **url** (*str | unicode*) – Partial URL path to send the request to. This will be joined to the end of the instance's base_uri attribute.
- **kwargs** (*dict*) – Additional keyword arguments to include in the requests call.

Returns The response of the request.

Return type requests.Response

post(*url*, ***kwargs*)
Performs a POST request.

Parameters

- **url** (*str | unicode*) – Partial URL path to send the request to. This will be joined to the end of the instance's base_uri attribute.
- **kwargs** (*dict*) – Additional keyword arguments to include in the requests call.

Returns The response of the request.

Return type requests.Response

put(*url*, ***kwargs*)
Performs a PUT request.

Parameters

- **url** (*str | unicode*) – Partial URL path to send the request to. This will be joined to the end of the instance's base_uri attribute.
- **kwargs** (*dict*) – Additional keyword arguments to include in the requests call.

Returns The response of the request.

Return type requests.Response

request(*method*, *url*, *headers=None*, ***kwargs*)
Main method for routing HTTP requests to the configured Vault base_uri. Intended to be implement by subclasses.

Parameters

- **method** (*str*) – HTTP method to use with the request. E.g., GET, POST, etc.
- **url** (*str | unicode*) – Partial URL path to send the request to. This will be joined to the end of the instance's base_uri attribute.
- **headers** (*dict*) – Additional headers to include with the request.
- **kwargs** (*dict*) – Additional keyword arguments to include in the requests call.

Returns The response of the request.

Return type requests.Response

static urljoin(**args*)
Joins given arguments into a url. Trailing and leading slashes are stripped for each argument.

Parameters **args** (*str | unicode*) – Multiple parts of a URL to be combined into one string.

Returns Full URL combining all provided arguments

Return type str | unicode

class hvac.adapters.**Request**(*base_uri='http://localhost:8200'*, *token=None*, *cert=None*, *verify=True*, *timeout=30*, *proxies=None*, *allow_redirects=True*, *session=None*)
Bases: *hvac.adapters.Adapter*

The Request adapter class

request(*method*, *url*, *headers=None*, ***kwargs*)
Main method for routing HTTP requests to the configured Vault base_uri.

Parameters

- **method** (*str*) – HTTP method to use with the request. E.g., GET, POST, etc.
- **url** (*str | unicode*) – Partial URL path to send the request to. This will be joined to the end of the instance's base_uri attribute.
- **headers** (*dict*) – Additional headers to include with the request.
- **kwargs** (*dict*) – Additional keyword arguments to include in the requests call.

Returns The response of the request.

Return type requests.Response

4.8 hvac.exceptions

exception hvac.exceptions.**Forbidden**(*message=None*, *errors=None*)
Bases: *hvac.exceptions.VaultError*

exception hvac.exceptions.**InternalServerError**(*message=None*, *errors=None*)
Bases: *hvac.exceptions.VaultError*

exception hvac.exceptions.**InvalidPath**(*message=None*, *errors=None*)
Bases: *hvac.exceptions.VaultError*

exception hvac.exceptions.**InvalidRequest**(*message=None*, *errors=None*)
Bases: *hvac.exceptions.VaultError*

exception hvac.exceptions.**ParamValidationError**(*message=None*, *errors=None*)
Bases: *hvac.exceptions.VaultError*

exception hvac.exceptions.**RateLimitExceeded**(*message=None*, *errors=None*)
Bases: *hvac.exceptions.VaultError*

exception hvac.exceptions.**Unauthorized**(*message=None*, *errors=None*)
Bases: *hvac.exceptions.VaultError*

exception hvac.exceptions.**UnexpectedError**(*message=None*, *errors=None*)
Bases: *hvac.exceptions.VaultError*

exception hvac.exceptions.**VaultDown**(*message=None*, *errors=None*)
Bases: *hvac.exceptions.VaultError*

exception hvac.exceptions.**VaultError**(*message=None*, *errors=None*)
Bases: exceptions.Exception

__init__(*message=None*, *errors=None*)
x.__init__(...) initializes x; see help(type(x)) for signature

exception hvac.exceptions.**VaultNotInitialized**(*message=None*, *errors=None*)
Bases: *hvac.exceptions.VaultError*

CHAPTER 5

Contributing

Feel free to open pull requests with additional features or improvements!

5.1 Testing

Integration tests will automatically start a Vault server in the background. Just make sure the latest `vault` binary is available in your `PATH`.

1. Install Vault or execute `VAULT_BRANCH=release scripts/install-vault-release.sh`
2. Install Tox
3. Run tests: `make test`

5.2 Documentation

5.2.1 Examples

Example code or general guides for methods in this module can be added under docs/usage. Any newly added or updated method in this module will ideally have a corresponding addition to these examples. New usage sections should also be added to the table of contents tracked in docs/usage.rst.

5.3 Backwards Compatibility Breaking Changes

Due to the close connection between this module and HashiCorp Vault versions, breaking changes are sometimes required. This can also occur as part of code refactoring to enable improvements in the module generally. In these cases:

- A deprecation notice should be displayed to callers of the module until the minor revision +2. E.g., a notice added in version 0.6.2 could see the marked method / functionality removed in version 0.8.0.

- Breaking changes should be called out in the CHANGELOG.md for the affected version.

5.4 Package Publishing Checklist

The follow list uses version number `0.6.2`, this string should be updated to match the intended release version. It is based on this document: https://gist.github.com/audreyr/5990987

- [ ] Ensure your working directory is clear by running: .. code-block:: guess

  make distclean

- [ ] Checkout a working branch: .. code-block:: guess

  git checkout -b master_v0-6-2

- [ ] Update CHANGELOG.md with a list of the included changes. Those changes can be reviewed, and their associated GitHub PR number confirmed, via GitHub's pull request diff using the previous version's tag. E.g.: https://github.com/hvac/hvac/compare/v0.6.1...master

- [ ] Commit the changelog changes: .. code-block:: guess

  git add CHANGELOG.md git commit -S -m "Updates for upcoming release 0.6.2"

- [ ] Update version number using bumpversion. This example is for the "patch" version but can also be "minor" or "major" as needed. .. code-block:: guess

  bumpversion patch version

- [ ] Commit the version changes: .. code-block:: guess

  git add version setup.cfg git commit -S -m "Bump patch version to $(cat version)"

- [ ] Install the package again for local development, but with the new version number: .. code-block:: guess

  python setup.py develop

- [ ] Run the tests and verify that they all pass: .. code-block:: guess

  make test

- [ ] Invoke setup.py / setuptools via the "package" Makefile job to create the release version's sdist and wheel artifacts:

```
make package
```

- [ ] Publish the sdist and wheel artifacts to TestPyPI using twine:

```
twine upload --repository-url https://test.pypi.org/legacy/ dist/*.tar.gz dist/*.
↪whl
```

- [ ] Check the TestPyPI project page to make sure that the README, and release notes display properly: https://test.pypi.org/project/hvac/

- [ ] Test that the version is correctly listed and it pip installs (`mktmpenv` is available via the virtualenvwrapper module) using the TestPyPI repository (Note: installation will currently fail due to missing recent releases of `requests` on TestPyPI): .. code-block:: guess

  mktmpenv pip install –no-cache-dir –index-url https://test.pypi.org/simple hvac== <verify releaes version shows up with the correct formatting in the resulting list> pip install –no-cache-dir –index-url https://test.pypi.org/simple hvac==0.6.2 <verify hvac functionality> deactivate

- [ ] Create a **draft** GitHub release using the contents of the new release version's CHANGELOG.md content: https://github.com/hvac/hvac/releases/new
- [ ] Upload the sdist and whl files to the draft GitHub release as attached "binaries".
- [ ] Push up the working branch (`git push`) and open a PR to merge the working branch into master: https://github.com/hvac/hvac/compare/master...master_v0-6-2
- [ ] After merging the working branch into master, tag master with the release version and push that up as well:

```
git checkout master
git pull
git tag "v$(cat version)"
git push "v$(cat version)"
```

- [ ] Publish the sdist and wheel artifacts to PyPI using twine:

```
twine upload dist/*.tar.gz dist/*.whl
```

- [ ] Check the PyPI project page to make sure that the README, and release notes display properly: https://pypi.org/project/hvac/
- [ ] Test that the version is correctly listed and it pip installs (`mktmpenv` is available via the virtualenvwrapper module) using the TestPyPI repository:

```
mktmpenv
pip install --no-cache-dir hvac==
<verify releaes version shows up with the correct formatting in the resulting␣
↪list>
pip install --no-cache-dir hvac==0.6.2
<verify hvac functionality>
deactivate
```

- [ ] Publish the draft release on GitHub: https://github.com/hvac/hvac/releases

CHAPTER 6

Changelog

6.1 0.6.4 (September 5th, 2018)

IMPROVEMENTS:

- New KV secret engine-related classes added. See the KV documentation under hvac's readthedocs.io site for usage / examples. GH-257 / GH-260

MISCELLANEOUS:

- Language classifiers are now being included with the distribution. GH-247
- Token no longer being sent in URL path for the `Client.renew_token` method. GH-250
- Support for the response structure in newer versions of Vault within the `Client.get_policy` method. GH-254
- `config` and `plugin_name` parameters added to the `Client.enable_auth_backend` method. GH-253

Thanks to @ijl, @rastut, @seuf, @downeast for their lovely contributions.

6.2 0.6.3 (August 8th, 2018)

DEPRECATION NOTICES:

- The `auth_github()` method within the `hvac.Client` class has been marked as deprecated and will be removed in hvac v0.8.0 (or later). Please update any callers of this method to use the `hvac.Client.github.login()` instead.
- The `auth_ldap()` method within the `hvac.Client` class has been marked as deprecated and will be removed in hvac v0.8.0 (or later). Please update any callers of this method to use the `hvac.Client.ldap.login()` instead.

IMPROVEMENTS:

- New Github auth method class added. See the Github documentation for usage / examples. GH-242

- New Ldap auth method class added. See the Ldap documentation for usage / examples. GH-244
- New Mfa auth method class added. See the documentation for usage / examples. GH-255
- `auth_aws_iam()` method updated to include "region" parameter for deployments in different AWS regions. GH-243

DOCUMENTATION UPDATES:

- Additional guidance for how to configure hvac's `Client` class to leverage self-signed certificates / private CA bundles has been added at: Making Use of Private CA. GH-230
- Docstring for `verify` `Client` parameter corrected and expanded. GH-238

MISCELLANEOUS:

- Automated PyPi deploys via travis-ci removed. GH-226
- Repository transferred to the new "hvac" GitHub organization; thanks @ianunruh! GH-227
- Codecov (automatic code coverage reports) added. GH-229 / GH-228
- Tests subdirectory reorganized; now broken up by integration versus unit tests with subdirectories matching the module path for the code under test. GH-236

Thanks to @otakup0pe, @FabianFrank, @andrewheald for their lovely contributions.

6.3 0.6.2 (July 19th, 2018)

BACKWARDS COMPATIBILITY NOTICE:

- With the newly added `hvac.adapters.Request` class, request kwargs can no longer be directly modified via the `_kwargs` attribute on the `Client` class. If runtime modifications to this dictionary are required, callers either need to explicitly pass in a new `adapter` instance with the desired settings via the `adapter` propery on the `Client` class *or* access the `_kwargs` property via the `adapter` property on the `Client` class.

See the Advanced Usage section of this module's documentation for additional details.

IMPROVEMENTS:

- sphinx documentation and readthedocs.io project added. GH-222
- README.md included in setuptools metadata. GH-222
- All `tune_secret_backend()` parameters now accepted. GH-215
- Add `read_lease()` method GH-218
- Added adapter module with `Request` class to abstract HTTP requests away from the `Client` class. GH-223

Thanks to @bbayszczak, @jvanbrunschot-coolblue for their lovely contributions.

6.4 0.6.1 (July 5th, 2018)

IMPROVEMENTS:

- Update `unwrap()` method to match current Vault versions [GH-149]
- Initial support for Kubernetes authentication backend [GH-210]
- Initial support for Google Cloud Platform (GCP) authentication backend [GH-206]
- Update enable_secret_backend function to support kv version 2 [GH-201]

BUG FIXES:

- Change URL parsing to allow for routes in the base Vault address (e.g., `https://example.com/vault`) [GH-212].

Thanks to @mracter, @cdsf, @SiN, @seanmalloy, for their lovely contributions.

6.5 0.6.0 (June 14, 2018)

BACKWARDS COMPATIBILITY NOTICE:

- Token revocation now sends the token in the request payload. Requires Vault >0.6.5
- Various methods have new and/or re-ordered keyword arguments. Code calling these methods with positional arguments may need to be modified.

IMPROVEMENTS:

- Ensure mount_point Parameter for All AWS EC2 Methods [GH-195]
- Add Methods for Auth Backend Tuning [GH-193]
- Customizable approle path / mount_point [GH-190]
- Add more methods for the userpass backend [GH-175]
- Add transit signature_algorithm parameter [GH-174]
- Add auth_iam_aws() method [GH-170]
- lookup_token function POST token not GET [GH-164]
- Create_role_secret_id with wrap_ttl & fix get_role_secret_id_accessor [GH-159]
- Fixed json() from dict bug and added additional arguments on auth_ec2() method [GH-157]
- Support specifying period when creating EC2 roles [GH-140]
- Added support for /sys/generate-root endpoint [GH-131] / [GH-199]
- Added "auth_cubbyhole" method [GH-119]
- Send token/accessor as a payload to avoid being logged [GH-117]
- Add AppRole delete_role method [GH-112]

BUG FIXES:

- Always Specify auth_type In create_ec2_role [GH-197]
- Fix "double parasing" of JSON response in auth_ec2 method [GH-181]

Thanks to @freimer, @ramiamar, @marcoslopes, @ianwestcott, @marc-sensenich, @sunghyun-lee, @jnaulty, @sijis, @Myles-Steinhauser-Bose, @oxmane, @ltm, @bchannak, @tkinz27, @crmulliner, for their lovely contributions.

6.6 0.5.0 (February 20, 2018)

IMPROVEMENTS:

- Added `disallowed_policies` parameter to `create_token_role` method [GH-169]

Thanks to @morganda for their lovely contribution.

6.7 0.4.0 (February 1, 2018)

IMPROVEMENTS:

- Add support for the `period` parameter on token creation [GH-167]
- Add support for the `cidr_list` parameter for approle secrets [GH-114]

BUG FIXES:

- Documentation is now more accurate [GH-165] / [GH-154]

Thanks to @ti-mo, @dhoeric, @RAbraham, @lhdumittan, @ahsanali for their lovely contributions.

6.8 0.3.0 (November 9, 2017)

This is just the highlights, there have been a bunch of changes!

IMPROVEVEMENTS:

- Some AppRole support [GH-77]
- Response Wrapping [GH-85]
- AWS EC2 stuff [GH-107], [GH-109]

BUG FIXES

- Better handling of various error states [GH-79], [GH-125]

Thanks to @ianwestcott, @s3u, @mracter, @intgr, @jkdihenkar, @gaelL, @henriquegemignani, @bfeeser, @nicr9, @mwielgoszewski, @mtougeron for their contributions!

6.9 0.2.17 (December 15, 2016)

IMPROVEMENTS:

- Add token role support [GH-94]
- Add support for Python 2.6 [GH-92]
- Allow setting the explicit_max_ttl when creating a token [GH-81]
- Add support for write response wrapping [GH-85]

BUG FIXES:

- Fix app role endpoints for newer versions of Vault [GH-93]

6.10 0.2.16 (September 12, 2016)

Thanks to @otakup0pe, @nicr9, @marcoslopes, @caiotomazelli, and @blarghmatey for their contributions!

IMPROVEMENTS:

- Add EC2 auth support [GH-61]
- Add support for token accessors [GH-69]

- Add support for response wrapping [GH-70]
- Add AppRole auth support [GH-77]

BUG FIXES:

- Fix `no_default_policy` parameter in `create_token` [GH-65]
- Fix EC2 auth double JSON parsing [GH-76]

6.11 0.2.15 (June 22nd, 2016)

Thanks to @blarghmatey, @stevenmanton, and @ahlinc for their contributions!

IMPROVEMENTS:

- Add methods for manipulating app/user IDs [GH-62]
- Add ability to automatically parse policies with pyhcl [GH-58]
- Add TTL option to `create_userpass` [GH-60]
- Add support for backing up keys on rekey [GH-57]
- Handle non-JSON error responses correctly [GH-46]

BUG FIXES:

- `is_authenticated` now handles new error type for Vault 0.6.0

6.12 0.2.14 (June 2nd, 2016)

BUG FIXES:

- Fix improper URL being used when leader redirection occurs [GH-56]

6.13 0.2.13 (May 31st, 2016)

IMPROVEMENTS:

- Add support for Requests sessions [GH-53]

BUG FIXES:

- Properly handle redirects from Vault server [GH-51]

6.14 0.2.12 (May 12th, 2016)

IMPROVEMENTS:

- Add support for `increment` in renewel of secret [GH-48]

BUG FIXES:

- Use unicode literals when constructing URLs [GH-50]

6.15 0.2.10 (April 8th, 2016)

IMPROVEMENTS:

- Add support for list operation [GH-47]

6.16 0.2.9 (March 18th, 2016)

IMPROVEMENTS:

- Add support for nonce during rekey operation [GH-42]
- Add get method for policies [GH-43]
- Add delete method for userpass auth backend [GH-45]
- Add support for response to rekey init

6.17 0.2.8 (February 2nd, 2016)

IMPROVEMENTS:

- Convenience methods for managing userpass and app-id entries
- Support for new API changes in Vault v0.4.0

6.18 0.2.7 (December 16th, 2015)

IMPROVEMENTS:

- Add support for PGP keys when rekeying [GH-28]

BUG FIXES:

- Fixed token metadata parameter [GH-27]

6.19 0.2.6 (October 30th, 2015)

IMPROVEMENTS:

- Add support for `revoke-self`
- Restrict `requests` dependency to modern version

6.20 0.2.5 (September 29th, 2015)

IMPROVEMENTS:

- Add support for API changes/additions in Vault v0.3.0
  - Tunable config on secret backends

  - MFA on username/password and LDAP auth backends
  - PGP encryption for unseal keys

6.21 0.2.4 (July 23rd, 2015)

BUG FIXES:

- Fix write response handling [GH-19]

6.22 0.2.3 (July 18th, 2015)

BUG FIXES

- Fix error handling for next Vault release

IMPROVEMENTS:

- Add support for rekey/rotate APIs

6.23 0.2.2 (June 12th, 2015)

BUG FIXES:

- Restrict `requests` dependency to 2.5.0 or later

IMPROVEMENTS:

- Return latest seal status from `unseal_multi`

6.24 0.2.1 (June 3rd, 2015)

BUG FIXES:

- Use arguments passed to `initialize` method

6.25 0.2.0 (May 25th, 2015)

BACKWARDS COMPATIBILITY NOTICE:

- Requires Vault 0.1.2 or later for `X-Vault-Token` header
- `auth_token` method removed in favor of `token` property
- `read` method no longer raises `hvac.exceptions.InvalidPath` on nonexistent paths

IMPROVEMENTS:

- Tolerate falsey URL in client constructor
- Add ability to auth without changing to new token
- Add `is_authenticated` convenience method

- Return `None` when reading nonexistent path

6.26 0.1.1 (May 20th, 2015)

IMPROVEMENTS:

- Add `is_sealed` convenience method
- Add `unseal_multi` convenience method

BUG FIXES:

- Remove secret_shares argument from `unseal` method

6.27 0.1.0 (May 17th, 2015)

- Initial release

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

h

Index

Symbols

A

C

D

E

F

G

H

I

K

L

M

P

R

S

T

U

V

W