python-barbicanclient Documentation

Release 5.3.1.dev4

OpenStack Foundation

May 18, 2022

CONTENTS

This is a client for OpenStack Key Management API (Barbican). Theres a Python API (the *barbicanclient* module), and a command-line interface (installed as *barbican*).

Contents:

CHAPTER

ONE

INSTALLATION

At the command line:

```
$ pip install python-barbicanclient
```

Or, if you have virtualenvwrapper installed:

```
$ mkvirtualenv python-barbicanclient
$ pip install python-barbicanclient
```

CHAPTER

TWO

USER DOCUMENTATION

2.1 CLI Usage

```
usage: barbican [--version] [-v] [--log-file LOG_FILE] [-q] [-h] [--debug]
                [--no-auth] [--os-identity-api-version <identity-api-version>]
                [--os-auth-url <auth-url>] [--os-username <auth-user-name>]
                [--os-user-id <auth-user-id>] [--os-password <auth-password>]
                [--os-user-domain-id <auth-user-domain-id>]
                [--os-user-domain-name <auth-user-domain-name>]
                [--os-tenant-name <auth-tenant-name>]
                [--os-tenant-id <tenant-id>]
                [--os-project-id <auth-project-id>]
                [--os-project-name <auth-project-name>]
                [--os-project-domain-id <auth-project-domain-id>]
                [--os-project-domain-name <auth-project-domain-name>]
                [--os-auth-token <auth-token>]
                [--endpoint <barbican-url>] [--insecure]
                [--os-cacert <ca-certificate>] [--os-cert <certificate>]
                [--os-key <key>] [--timeout <seconds>]
```

The examples below assume that credentials have been saved to your environment. If you dont have variables saved to your environment or you wish to use different credentials than those defined, any of the optional arguments listed above may be passed to Barbican.

Barbican takes a positional argument <entity>, which specifies whether you wish to operate on a secret or an order.

2.1.1 Secrets

```
$ barbican secret <action>
```

A subcommand describing the action to be performed should follow. The subcommands are mostly the same for secrets and orders, although some optional arguments only apply to one or the other.

Subcommand actions that a user can take for secrets are:

```
secret delete  Delete a secret by providing its URI.
secret get     Retrieve a secret by providing its URI.
```

(continues on next page)

(continued from previous page)

```
secret list    List secrets.
secret store   Store a secret in Barbican.
```

Each subcommand takes in a different set of arguments, and the help message varies from one to another. The help message for **get** can be seen below.

```
$  barbican help secret get
usage: barbican secret get [-h] [-f {json,shell,table,value,yaml}] [-c COLUMN]
                           [--max-width <integer>] [--fit-width]
                           [--print-empty] [--noindent] [--prefix PREFIX]
                           [--decrypt | --payload | --file <filename>]
                           [--payload_content_type PAYLOAD_CONTENT_TYPE]
                           URI

Retrieve a secret by providing its URI.

positional arguments:
  URI                   The URI reference for the secret.

optional arguments:
  -h, --help            show this help message and exit
  --decrypt, -d         if specified, retrieve the unencrypted secret data.
  --payload, -p         if specified, retrieve the unencrypted secret data.
  --file <filename>, -F <filename>
                        if specified, save the payload to a new file with the
                        given filename.
  --payload_content_type PAYLOAD_CONTENT_TYPE, -t PAYLOAD_CONTENT_TYPE
                        the content type of the decrypted secret (default:
                        text/plain).

output formatters:
  output formatter options

  -f {shell,table,value}, --format {shell,table,value}
                        the output format, defaults to table
  -c COLUMN, --column COLUMN
                        specify the column(s) to include, can be repeated

table formatter:
  --max-width <integer>
                        Maximum display width, 0 to disable

shell formatter:
  a format a UNIX shell can parse (variable="value")

  --prefix PREFIX       add a prefix to all variable names
```

Secret Create

```
$ barbican secret store -n mysecretname -p 'my secret value'

+---------------+------------------------------------------------------------
↪----------+
| Field         | Value                                                      ␣
↪          |
+---------------+------------------------------------------------------------
↪----------+
| Secret href   | http://localhost:9311/v1/secrets/a70a45d8-4076-42a2-b111-
↪8893d3b92a3e |
| Name          | mysecretname                                               ␣
↪          |
| Created       | None                                                       ␣
↪          |
| Status        | None                                                       ␣
↪          |
| Content types | None                                                       ␣
↪          |
| Algorithm     | aes                                                        ␣
↪          |
| Bit length    | 256                                                        ␣
↪          |
| Mode          | cbc                                                        ␣
↪          |
| Expiration    | None                                                       ␣
↪          |
+---------------+------------------------------------------------------------
↪----------+
```

Instead of using the `-p` or `--payload` option with the value of the secret in the command line, the value of the secret may be stored in a file. For this method the `-F <filename>` or `--file <filename>` option can be used.

Secret Get

```
$ barbican secret get http://localhost:9311/v1/secrets/a70a45d8-4076-42a2-
↪b111-8893d3b92a3e

+---------------+------------------------------------------------------------
↪----------+
| Field         | Value                                                      ␣
↪          |
+---------------+------------------------------------------------------------
↪----------+
| Secret href   | http://localhost:9311/v1/secrets/a70a45d8-4076-42a2-b111-
↪8893d3b92a3e |
| Name          | mysecretname                                               ␣
↪          |
```

(continues on next page)

(continued from previous page)

```
| Created        | 2015-04-16 20:36:40.334696+00:00                                ␣
↪               |
| Status         | ACTIVE                                                          ␣
↪               |
| Content types  | {u'default': u'application/octet-stream'}                       ␣
↪               |
| Algorithm      | aes                                                             ␣
↪               |
| Bit length     | 256                                                             ␣
↪               |
| Mode           | cbc                                                             ␣
↪               |
| Expiration     | None                                                            ␣
↪               |
+----------------+-----------------------------------------------------------------
↪----------+
```

To retrieve only the raw value of the payload we have introduced the `-p` or `--payload` option paired with the `-f value` cliff formatting option. (The `--decrypt` option will perform the same action; however, it will be deprecated)

```
$ barbican secret get http://localhost:9311/v1/secrets/a70a45d8-4076-42a2-
↪b111-8893d3b92a3e --payload -f value
my secret value
```

Instead of using the `-p` or `--payload` option with the value of the secret returned to stdout, the value of the secret may be written to a file. For this method the `-F <filename>` or `--file <filename>` option can be used.

Secret Delete

```
$ barbican secret delete http://localhost:9311/v1/secrets/a70a45d8-4076-42a2-
↪b111-8893d3b92a3e
```

Secret Update

```
$ barbican secret update http://localhost:9311/v1/secrets/a70a45d8-4076-42a2-
↪b111-8893d3b92a3e ``my_payload``
```

In order for a secret to be updated it must have been created without a payload. `my_payload` will be added as the secrets payload.

Secret List

```
$ barbican secret list

+------------------------------------------------------------------------+-----
↪-+--------------------------------+--------+-----------------------------
↪-------------+-----------+------------+------+------------+
| Secret href                                                            |␣
↪Name | Created                        | Status | Content types           ␣
↪                  | Algorithm | Bit length | Mode | Expiration |
+------------------------------------------------------------------------+-----
↪-+--------------------------------+--------+-----------------------------
↪-------------+-----------+------------+------+------------+
| http://localhost:9311/v1/secrets/bb3d8c20-8ea5-4bfc-9645-c8da79c8b371 |␣
↪None | 2015-04-15 20:37:37.501475+00:00 | ACTIVE | {u'default': u
↪'application/octet-stream'} | aes       |        256 | cbc  | None       |
+------------------------------------------------------------------------+-----
↪-+--------------------------------+--------+-----------------------------
↪-------------+-----------+------------+------+------------+
```

2.1.2 ACLS

```
$ barbican acl <action>
```

A subcommand describing the action to be performed should follow. The subcommands are mostly the same for secret and container ACLs.

Subcommand actions that a user can take for ACLs are:

```
acl delete          Delete ACLs for a secret or container as identified by␣
↪its href.
acl get             Retrieve ACLs for a secret or container by providing its␣
↪href.
acl submit          Submit ACL on a secret or container as identified by its␣
↪href.
acl user add        Add ACL users to a secret or container as identified by␣
↪its href.
acl user remove     Remove ACL users from a secret or container as identified␣
↪by its href.
```

ACL **get** or **delete** subcommand, only takes secret or container href. All other ACL commands take additional arguments to specify ACL settings data. Please see help message for both cases of argument. Either secret ref or container ref is required for all of acl actions.

```
$ barbican help acl get
usage: barbican acl get [-h] [-f {csv,table,value}] [-c COLUMN]
                        [--max-width <integer>]
                        [--quote {all,minimal,none,nonnumeric}]
                        URI
```

(continues on next page)

(continued from previous page)

```
Retrieve ACLs for a secret or container by providing its href.

positional arguments:
  URI                   The URI reference for the secret or container.

optional arguments:
  -h, --help            show this help message and exit

output formatters:
  output formatter options

  -f {csv,table,value}, --format {csv,table,value}
                        the output format, defaults to table
  -c COLUMN, --column COLUMN
                        specify the column(s) to include, can be repeated

table formatter:
  --max-width <integer>
                        Maximum display width, 0 to disable

CSV Formatter:
  --quote {all,minimal,none,nonnumeric}
                        when to include quotes, defaults to nonnumeric
```

Following is snippet of related command line options for an ACL modify action e.g. submit, add or remove.

```
$ barbican help acl submit/user add/user remove
usage: barbican acl submit [-h] [-f {csv,table,value}] [-c COLUMN]
                           [--max-width <integer>]
                           [--quote {all,minimal,none,nonnumeric}]
                           [--user [USER]]
                           [--project-access | --no-project-access]
                           [--operation-type {read}]
                           URI

....
....

positional arguments:
  URI                   The URI reference for the secret or container.

optional arguments:
  -h, --help            show this help message and exit
  --user [USER], -u [USER]
                        Keystone userid(s) for ACL.
  --project-access      Flag to enable project access behavior.
  --no-project-access   Flag to disable project access behavior.
  --operation-type {read}, -o {read}
```

(continues on next page)

(continued from previous page)

```
                        Type of Barbican operation ACL is set for
....
....
```

Note: Default for `operation-type` argument is read as thats the only operation currently supported by Barbican ACL API. So this argument can be skipped in CLI call.

ACLs Get

To get complete ACL setting for a secret or container, use this ACL action.

```
$ barbican acl get http://localhost:9311/v1/secrets/7776adb8-e865-413c-8ccc-
↪4f09c3fe0213

+---------------+---------------+----------------------------------------
↪--------------------------------+--------------------------------+-------
↪--------------------------+----------------------------------------------
↪--------------------------+
| Operation Type | Project Access | Users                                  ␣
↪                                | Created                        |␣
↪Updated                        | Secret ACL Ref                         ␣
↪                                |
+---------------+---------------+----------------------------------------
↪--------------------------------+--------------------------------+-------
↪--------------------------+----------------------------------------------
↪--------------------------+
| read           | False          | [u'721e27b8505b499e8ab3b38154705b9e', u
↪'2d0ee7c681cc4549b6d76769c320d91f'] | 2015-07-21 17:52:01.729370+00:00 |␣
↪2015-07-28 02:08:02.455276+00:00 | http://localhost:9311/v1/secrets/
↪7776adb8-e865-413c-8ccc-4f09c3fe0213/acl |
+---------------+---------------+----------------------------------------
↪--------------------------------+--------------------------------+-------
↪--------------------------+----------------------------------------------
↪--------------------------+

$ barbican acl get http://localhost:9311/v1/containers/83c302c7-86fe-4f07-
↪a277-c4962f121f19

+---------------+---------------+----------------------------------------+---
↪--------------------------------+--------------------------------+---------
↪-------------------------------------------------------------------+
| Operation Type | Project Access | Users                                  |␣
↪Created                        | Updated                        |␣
↪Container ACL Ref                                                      ␣
↪ |
+---------------+---------------+----------------------------------------+---
↪--------------------------------+--------------------------------+---------
↪-------------------------------------------------------------------+
```

(continues on next page)

(continued from previous page)

```
| read           | False          | [u'2d0ee7c681cc4549b6d76769c320d91f'] |␣
↪2015-07-28 01:36:55.791381+00:00 | 2015-07-28 02:05:41.175386+00:00 | http:/
↪/localhost:9311/v1/containers/83c302c7-86fe-4f07-a277-c4962f121f19/acl |
+----------------+----------------+---------------------------------------+---
↪-------------------------------+-----------------------------------+---------
↪-----------------------------------------------------------------+
```

Secret or container ref is required. If missing, it will result in error.

```
$ barbican acl get

usage: barbican acl get [-h] [-f {csv,table,value}] [-c COLUMN]
                        [--max-width <integer>]
                        [--quote {all,minimal,none,nonnumeric}]
                        URI
barbican acl get: error: too few arguments
```

ACLs Submit

To submit complete ACL setting for a secret or container, use this ACL action.

```
$ barbican acl submit --user 2d0ee7c681cc4549b6d76769c320d91f --user␣
↪721e27b8505b499e8ab3b38154705b9e http://localhost:9311/v1/secrets/7776adb8-
↪e865-413c-8ccc-4f09c3fe0213

+----------------+----------------+---------------------------------------
↪-------------------------------+-----------------------------------+-------
↪--------------------------+-----------------------------------------------
↪--------------------------+
| Operation Type | Project Access | Users                                 ␣
↪                               | Created                           |␣
↪Updated                        | Secret ACL Ref                        ␣
↪                               |
+----------------+----------------+---------------------------------------
↪-------------------------------+-----------------------------------+-------
↪--------------------------+-----------------------------------------------
↪--------------------------+
| read           | True           | [u'721e27b8505b499e8ab3b38154705b9e', u
↪'2d0ee7c681cc4549b6d76769c320d91f'] | 2015-07-21 17:52:01.729370+00:00 |␣
↪2015-08-12 09:53:20.225971+00:00 | http://localhost:9311/v1/secrets/
↪7776adb8-e865-413c-8ccc-4f09c3fe0213/acl |
+----------------+----------------+---------------------------------------
↪-------------------------------+-----------------------------------+-------
↪--------------------------+-----------------------------------------------
↪--------------------------+
```

If `user` argument is missing or has no value, then empty list is passed for users and this approach can be used to remove existing ACL users. If project access argument is not provided, then by default project access is enabled. To disable project access behavior, just pass `no-project-access` argument without

any value.

```
$ barbican acl submit --user --no-project-access http://localhost:9311/v1/
↪secrets/7776adb8-e865-413c-8ccc-4f09c3fe0213

+----------------+----------------+-------+----------------------------------
↪+--------------------------------+----------------------------------------
↪--------------------------------+
| Operation Type | Project Access | Users | Created                          ␣
↪| Updated                        | Secret ACL Ref                           ␣
↪                                |
+----------------+----------------+-------+----------------------------------
↪+--------------------------------+----------------------------------------
↪--------------------------------+
| read           | False          | []    | 2015-07-21 17:52:01.729370+00:00␣
↪| 2015-08-12 09:55:23.043433+00:00 | http://localhost:9311/v1/secrets/
↪7776adb8-e865-413c-8ccc-4f09c3fe0213/acl |
+----------------+----------------+-------+----------------------------------
↪+--------------------------------+----------------------------------------
↪--------------------------------+

$ barbican acl submit --user 2d0ee7c681cc4549b6d76769c320d91f --no-project-
↪access http://localhost:9311/v1/containers/83c302c7-86fe-4f07-a277-
↪c4962f121f19

+----------------+----------------+--------------------------------------+---
↪-------------------------------+----------------------------------+---------
↪-----------------------------------------------------------------+
| Operation Type | Project Access | Users                                |␣
↪Created                        | Updated                          |␣
↪Container ACL Ref                                                        ␣
↪ |
+----------------+----------------+--------------------------------------+---
↪-------------------------------+----------------------------------+---------
↪-----------------------------------------------------------------+
| read           | False          | [u'2d0ee7c681cc4549b6d76769c320d91f'] |␣
↪2015-07-29 22:01:00.878270+00:00 | 2015-08-19 05:56:09.930302+00:00 | http:/
↪/localhost:9311/v1/containers/83c302c7-86fe-4f07-a277-c4962f121f19/acl |
+----------------+----------------+--------------------------------------+---
↪-------------------------------+----------------------------------+---------
↪-----------------------------------------------------------------+
```

Following error is returned when both mutually exclusive flags are passed.

```
$ barbican acl submit --project-access --no-project-access http://
↪localhost:9311/v1/secrets/7776adb8-e865-413c-8ccc-4f09c3fe0213
usage: barbican acl submit [-h] [-f {csv,table,value}] [-c COLUMN]
                           [--max-width <integer>]
                           [--quote {all,minimal,none,nonnumeric}]
                           [--user [USER]]
                           [--project-access | --no-project-access]
```

(continues on next page)

(continued from previous page)

```
                          [--operation-type {read}]
                          URI
barbican acl submit: error: argument --no-project-access: not allowed with␣
↪argument --project-access
```

ACL Add User(s)

To add ACL users for a secret or container, use this ACL action.

If `user` argument is missing or has no value, then no change is made in ACL users. If project access argument is not provided, then no change is made in existing project access behavior flag.

```
$ barbican acl user add --user c1d20e4b7e7d4917aee6f0832152269b http://
↪localhost:9311/v1/containers/83c302c7-86fe-4f07-a277-c4962f121f19

+---------------+---------------+-----------------------------------------
↪----------------------------------+-----------------------------------+-------
↪---------------------------+-----------------------------------------------
↪------------------------------+
| Operation Type | Project Access | Users                                  ␣
↪                                | Created                           |␣
↪Updated                         | Container ACL Ref                      ␣
↪                                  |
+---------------+---------------+-----------------------------------------
↪----------------------------------+-----------------------------------+-------
↪---------------------------+-----------------------------------------------
↪------------------------------+
| read          | False         | [u'2d0ee7c681cc4549b6d76769c320d91f', u
↪'c1d20e4b7e7d4917aee6f0832152269b'] | 2015-07-29 22:01:00.878270+00:00 |␣
↪2015-08-12 10:08:19.129370+00:00 | http://localhost:9311/v1/containers/
↪83c302c7-86fe-4f07-a277-c4962f121f19/acl |
+---------------+---------------+-----------------------------------------
↪----------------------------------+-----------------------------------+-------
↪---------------------------+-----------------------------------------------
↪------------------------------+
```

```
# Added 2 users for secret (084c2098-66db-4401-8348-d969be0eddaa) earlier via␣
↪set action.
$ barbican acl user add --user --no-project-access http://localhost:9311/v1/
↪secrets/084c2098-66db-4401-8348-d969be0eddaa

+---------------+---------------+-----------------------------------------
↪----------------------------------+-----------------------------------+-------
↪---------------------------+-----------------------------------------------
↪------------------------------+
| Operation Type | Project Access | Users                                  ␣
↪                                | Created                           |␣
↪Updated                         | Secret ACL Ref                         ␣
↪                                  |
```

(continues on next page)

(continued from previous page)

```
+---------------+---------------+-----------------------------------------
↪--------------------------------+--------------------------------+-------
↪--------------------------+-----------------------------------------
↪--------------------------+
| read          | False         | [u'721e27b8505b499e8ab3b38154705b9e', u
↪'2d0ee7c681cc4549b6d76769c320d91f'] | 2015-08-12 10:09:27.564371+00:00 |␣
↪2015-08-12 10:11:09.749980+00:00 | http://localhost:9311/v1/secrets/
↪084c2098-66db-4401-8348-d969be0eddaa/acl |
+---------------+---------------+-----------------------------------------
↪--------------------------------+--------------------------------+-------
↪--------------------------+-----------------------------------------
↪--------------------------+
```

ACL Remove User(s)

To remove ACL users for a secret or container, use this ACL action.

If `user` argument is missing or has no value, then no change is made in ACL users. If project access argument is not provided, then no change is made in existing project access behavior flag.

If provided userid(s) does not exist in ACL, it is simply ignored and only existing userid(s) are removed from ACL.

```
$ barbican acl user remove --user 2d0ee7c681cc4549b6d76769c320d91f --user␣
↪invalid_user_id http://localhost:9311/v1/secrets/084c2098-66db-4401-8348-
↪d969be0eddaa

+---------------+---------------+-----------------------------------------+---
↪--------------------------------+--------------------------------+---------
↪-----------------------------------------------------------------+
| Operation Type | Project Access | Users                                   |␣
↪Created                        | Updated                         |␣
↪Secret ACL Ref                                                          |
+---------------+---------------+-----------------------------------------+---
↪--------------------------------+--------------------------------+---------
↪-----------------------------------------------------------------+
| read          | False         | [u'721e27b8505b499e8ab3b38154705b9e'] |␣
↪2015-08-12 10:09:27.564371+00:00 | 2015-08-12 10:12:21.842888+00:00 | http:/
↪/localhost:9311/v1/secrets/084c2098-66db-4401-8348-d969be0eddaa/acl |
+---------------+---------------+-----------------------------------------+---
↪--------------------------------+--------------------------------+---------
↪-----------------------------------------------------------------+
```

ACLs Delete

To delete existing ACL setting for a secret or container, use this ACL action.

```
$ barbican acl delete http://localhost:9311/v1/secrets/084c2098-66db-4401-
↪8348-d969be0eddaa

$ barbican acl get http://localhost:9311/v1/secrets/084c2098-66db-4401-8348-
↪d969be0eddaa

+----------------+----------------+-------+---------+---------+--------------
↪-------------------------------------------------------------+
| Operation Type | Project Access | Users | Created | Updated | Secret ACL␣
↪Ref                                                          |
+----------------+----------------+-------+---------+---------+--------------
↪-------------------------------------------------------------+
| read           | True           | []    | None    | None    | http://
↪localhost:9311/v1/secrets/084c2098-66db-4401-8348-d969be0eddaa/acl |
+----------------+----------------+-------+---------+---------+--------------
↪-------------------------------------------------------------+
```

2.2 Authentication

2.2.1 Keystone Authentication

The client defers authentication to Keystone Sessions, which provide several authentication plugins in the *keystoneauth1.identity* namespace. Below we give examples of the most commonly used auth plugins.

Keystone API Version 3 Authentication

Authentication using Keystone API Version 3 can be achieved using the *keystoneauth1.identity.V3Password* auth plugin.

Example:

```
from barbicanclient import client
from keystoneauth1 import identity
from keystoneauth1 import session

auth = identity.V3Password(auth_url='http://localhost:5000/v3',
                           username='admin_user',
                           user_domain_name='Default',
                           password='password',
                           project_name='demo',
                           project_domain_name='Default')
sess = session.Session(auth=auth)
barbican = client.Client(session=sess)
```

Keystone API Version 2 Authentication

Authentication using Keystone API Version 2 can be achieved using the *keystoneauth1.identity.V2Password* auth plugin.

Example:

```
from barbicanclient import client
from keystoneauth1 import identity
from keystoneauth1 import session

auth = identity.V2Password(auth_url='http://localhost:5000/v2.0',
                           username='admin_user',
                           password='password',
                           tenant_name='demo')
sess = session.Session(auth=auth)
barbican = client.Client(session=sess)
```

2.2.2 Unauthenticated Context

Sometimes it may be useful to work with the client in an unauthenticated context, for example when using a development instance of Barbican that is not yet configured to use Keystone for authentication. In this case, the Barbican Service endpoint must be provided, in addition to the Project ID that will be used for context (i.e. the project that owns the secrets youll be working with).

Example:

```
from barbicanclient import client

barbican = client.Client(endpoint='http://localhost:9311',
                         project_id='123456')
```

2.3 CLI Authentication

2.3.1 Keystone V3 Authentication

Barbican can be configured to use Keystone for authentication. The users credentials can be passed to Barbican via arguments.

```
$ barbican --os-auth-url <keystone-v3-url> --os-project-domain-id \
<domain id> --os-user-domain-id <user domain id> --os-username <username> \
--os-password <password> --os-project-name <project-name> secret list
```

This can become annoying and tedious, so authentication via Keystone can also be configured by setting environment variables. Barbican uses the same env variables as python-keystoneclient so if you already have keystone client configured you can skip this section.

An example clientrc file is provided in the main python-barbicanclient directory.

```
export OS_PROJECT_NAME=<YourProjectName>

# Either Project Domain ID or Project Domain Name is required
export OS_PROJECT_DOMAIN_ID=<YourProjectDomainID>
export OS_PROJECT_DOMAIN_NAME=<YourProjectDomainName>

# Either User Domain ID or User Domain Name is required
export OS_USER_DOMAIN_ID=<YourUserDomainID>
export OS_USER_DOMAIN_NAME=<YourUserDomainName>

# Either User ID or Username can be used
export OS_USER_ID =<YourUserID>
export OS_USERNAME=<YourUserName>

export OS_PASSWORD=<YourPassword>

# OS_AUTH_URL should be your location of Keystone
# Barbican Client defaults to Keystone V3
export OS_AUTH_URL="<YourAuthURL>:5000/v3/"
export BARBICAN_ENDPOINT="<YourBarbicanEndpoint>:9311"
```

Make any appropriate changes to this file.

You will need to source it into your environment on each load:

```
source ~/clientrc
```

If you would like, you can configure your bash to load the variables on each login:

```
echo "source ~/clientrc" >> ~/.bashrc
```

2.3.2 Keystone Token Authentication

Barbican can be configured to use Keystone tokens for authentication. The users credentials can be passed to Barbican via arguments.

```
$ barbican --os-auth-url <auth_endpoint> --os-auth-token <auth_token> \
--os-project-id <project_id> secret list
```

Much like normal password authentication you can specify these values via environmental variables. Refer to *Keystone V3 authentication* for more information.

2.3.3 No Auth Mode

When working with a Barbican instance that does not use Keystone authentication (e.g. during development) you can use the `--no-auth` option. If you do this, youll have to specify the Barbican endpoint and project ID `--os-project-id`. This is because Barbican normally gets the endpoint and tenant ID from Keystone.

2.4 Client Usage

To use barbicanclient, you must first create an instance of the *barbicanclient.client.Client* class.

The client uses Keystone Sessions for both authentication and for handling HTTP requests. You can provide authentication credentials to the client by creating a Keystone Session with the appropriate auth plugin and then passing that session to the new Client.

See *Authentication* for more details.

Example:

```
from barbicanclient import client

barbican = client.Client(...)
```

The client object has different attributes that can be used to interact with the Barbican service. Each attribute represents an entity in the Barbican service: Secrets, Orders and Containers.

2.4.1 Secrets

Secrets represent keys, credentials, and other sensitive data that is stored by the Barbican service. To store or retrieve a secret in the Barbican service you should use the different methods of the `barbicanclient.secrets.SecretManager` class that is exposed as the *secrets* attribute of the Client.

Example:

```
# Store a random text password in Barbican

from barbicanclient import client
import random
import string

def random_password(length):
    sys_random = random.SystemRandom()
    return u''.join(
        sys_random.choice(string.ascii_letters + string.digits) for _
↪ in range(length)
    )

barbican = client.Client(...)

my_secret = barbican.secrets.create()
my_secret.name = u'Random plain text password'
```

(continues on next page)

(continued from previous page)

```
my_secret.payload = random_password(24)

my_secret_ref = my_secret.store()
```

The secret reference returned by `barbicanclient.secrets.SecretManager.store()` can later be used to retrieve the secret data from barbican.

Example:

```
# Retrieve Secret from secret reference

retrieved_secret = barbican.secrets.get(my_secret_ref)
my_password = retrieved_secret.payload
```

Secret Content Types

The Barbican service defines a Secret Content Type. The client will choose the correct Content Type based on the type of the data that is set on the *Secret.payload* property. The following table summarizes the mapping of Python types to Barbican Secret Content Types:

six Type	Python 2 Type	Python 3 Type	Barbican Content Type
six.binary_type	str	bytes	application/octet-stream
six.text_type	unicode	str	text/plain

Warning: Previous versions of python-barbicanclient allowed the user to set the *payload_content_type* and *payload_content_encoding* properties for any secret. This can lead to unexpected behavior such as changing a unicode string back to a byte string in Python 2, and dropping the base64 encoding of a binary secret as in Launchpad Bug #1419166. Because of this, manually setting the *payload_content_type* and the *payload_content_encoding* has been deprecated.

2.4.2 Orders

Orders are used to request secret material to be created by the Barbican service. Submitting an order will result in a Secret being created on your behalf. The Secret can then be used like any Secret you may have uploaded yourself. Orders should be created using the factory methods in the `barbicanclient.orders.OrderManager` instance in the *orders* attribute of the *Client*.

Example:

```
# Submit an order to generate a random encryption key

from barbicanclient import client

barbican = client.Client(...)

my_order = barbican.orders.create_key()
```

(continues on next page)

(continued from previous page)

```
my_order.algorithm = 'AES'
my_order.mode = 'CBC'
my_order.bit_length = 256

my_order_ref = my_order.submit()
```

The order reference returned by `barbicanclient.orders.Order.submit()` can later be used to retrieve the order from Barbican.

Example:

```
# Retrieve Order from order reference

retrieved_order = barbican.orders.get(my_order_ref)
```

Once your order has been processed by Barbican, the order status will be set to *ACTIVE*. An active order will contain the reference to the requested secret (or container).

Example:

```
# Retrieve Encryption Key generated by the above KeyOrder

generated_secret = barbican.secrets.get(retrieved_order.secret_ref)
key = generated_secret.payload
```

Currently the client can submit `barbicanclient.orders.KeyOrder` orders for Keys suitable for symmetric encryption, and `barbicanclient.orders.AsymmetricOrder` for Asymmetric keys such as RSA keys.

2.4.3 Containers

Containers can be either arbitrary groupings of *Secrets* or a strict grouping of Secrets, such as the Public and Private keys of an RSA keypair.

Containers should be managed using the `barbicanclient.containers.ContainerManager` instance in the *containers* attribute of the *Client*

Example:

```
# Add the Secrets created above to a container

my_container = barbican.containers.create()

my_container.add('Retrieved Secret', retrieved_secret)
my_container.add('Generated Secret', generated_secret)

my_container_ref = my_container.store()
```

The container reference returned by `barbicanclient.containers.Container.store()` can later be used to retrieve the container from Barbican.

Example:

```
# Retrieve container from Barbican

retrieved_container = barbican.containers.get(my_container_ref)
```

2.4.4 ACLs

Access Control List (ACL) feature in Barbican provides user level access control for secrets and containers. By default Barbican manages access to its resources (secrets, containers) on a per project level and authorization is granted based on the roles a user has in that project.

ACLs should be managed using the `barbicanclient.acls.ACLManager` instance in the *acls* attribute of the *Client*.

Example:

```
# Submits ACLs on an existing Secret with URI as 'secret_ref'

# create ACL entity object with needed settings
acl_entity = barbican.acls.create(entity_ref=secret_ref, users=[u1,␣
↪u2],
                                  project_access=False)

acl_ref = acl_entity.submit()  # submits ACL setting to server at␣
↪this point.
```

The secret or container URI can be used to read all of its ACL setting. Returned value is instance of either `barbicanclient.acls.SecretACL` or `barbicanclient.acls.ContainerACL`. Refer to respective class for its available APIs.

Example:

```
# Get ACL entity for a Secret
# Returned entity will be either SecretACL or ContainerACL.
# This entity has ACL settings per operation type (e.g. 'read')

secret_acl = barbican.acls.get(secret_ref)

# To retrieve (load) ACL using existing ACL entity e.g. container_acl
container_acl.load_acls_data()
```

ACLs setting can also be retrieved directly from secret or container entity. Its data is lazy loaded i.e. related ACL settings are not read till *acls* attribute is accessed on secret or container entity.

Example:

```
# Get secret entity for a given ref
secret = barbican.secrets.get(secret_ref)

# To get project access flag or users for 'read' operation
project_access_flag = secret.acls.read.project_access
read_acl_users = secret.acls.read.users
```

(continues on next page)

(continued from previous page)

```
# Get container entity for a given ref
container = barbican.containers.get(container_ref)

# To get project access flag or users for 'read' operation
project_access_flag = container.acls.read.project_access
read_acl_users = container.acls.read.users
```

If need to add users to existing read ACL settings on a secret or container, above mentioned get and submit methods can be used.

Example:

```
# Every Barbican secret and container has default ACL setting which
# reflects default project access behavior.

# ACL settings is modified via submit operation on ACL entity.

# provide users to be added as list.
add_users = ['user1', 'user2', 'users3']

# Case 1 - Add users to 'read' operation ACL setting
# -------------------------------------------------

# Get ACL entity from server
acl_entity = barbican.acls.get(entity_ref=secret_ref)

# add new users to existing users for 'read' operation
acl_entity.read.users.extend(add_users)
# OR
# acl_entity.get('read').users.extend(add_users)

acl_ref = acl_entity.submit() # here submits ACL changes to server.

# Case 2 - Add same users to ACL settings for each operation type
# ---------------------------------------------------------------

# Get ACL entity from server
acl_entity = barbican.acls.get(entity_ref=secret_ref)

# Go through each operation ACL setting and add users to existing␣
↪list
for op_acl in acl_entity.operation_acls
    op_acl.users.extend(add_users)

acl_ref = acl_entity.submit() # here submits ACL changes to server.
```

If need to remove some users from existing ACL settings on a secret or container, similar approach can be used as mentioned above for *add* example.

Example:

```
# provide users to be removed as list.
remove_users = ['user1', 'user2', 'users3']

# Case 1 - Remove users from 'read' operation ACL setting
# ------------------------------------------------------

# Get ACL entity from server
acl_entity = barbican.acls.get(entity_ref=container_ref)

existing_users = acl_entity.read.users
# OR
# existing users = acl_entity.get('read').users

# remove matching users from existing users list
updated_users = set(existing_users).difference(remove_users)

# set back updated users to operation specific acl setting
acl_entity.read.users = updated_users
# OR
# acl_entity.get('read').users = updated_users

acl_ref = acl_entity.submit() # here submits ACL changes to server.

# Case 2 - Remove same users from ACL settings for each operation␣
↪type
# ------------------------------------------------------------------
↪-

# Get ACL entity from server
acl_entity = barbican.acls.get(secret_ref)

# Go through each operation ACL setting and remove users from␣
↪existing list
for op_acl in acl_entity.operation_acls
    existing_users = op_acl.users

    # remove matching users from existing users list
    updated_users = set(existing_users).difference(remove_users)

    # set back updated users to operation specific acl setting
    op_acl.users = updated_users

acl_ref = acl_entity.submit() # here submits ACL changes to server.
```

If need to unset or delete ACL settings on a secret or container, `barbicanclient.acls.SecretACL.remove()` or `barbicanclient.acls.ContainerACL.remove()` can be used.

Example:

```
# create ACL entity object with secret or container ref
blank_acl_entity = barbican.acls.create(entity_ref=secret_ref)

# removes all ACL settings for the secret on server
blank_acl_entity.remove()

# To remove 'read' operation specific ACL setting
acl_entity = barbican.acls.get(entity_ref=secret_ref)
acl_entity.read.remove()
# OR
# acl_entity.get('read').remove()
```

2.5 Key Manager service (barbican) command-line client

The barbican client is the command-line interface (CLI) for the Key Manager service (barbican) API and its extensions.

This chapter documents **barbican** version `4.3.0`.

For help on a specific **barbican** command, enter:

```
$ barbican help COMMAND
```

2.5.1 barbican usage

```
usage: barbican [--version] [-v | -q] [--log-file LOG_FILE] [-h] [--debug]
                [--no-auth] [--os-identity-api-version <identity-api-version>]
                [--os-auth-url <auth-url>] [--os-username <auth-user-name>]
                [--os-user-id <auth-user-id>] [--os-password <auth-password>]
                [--os-user-domain-id <auth-user-domain-id>]
                [--os-user-domain-name <auth-user-domain-name>]
                [--os-tenant-name <auth-tenant-name>]
                [--os-tenant-id <tenant-id>]
                [--os-project-id <auth-project-id>]
                [--os-project-name <auth-project-name>]
                [--os-project-domain-id <auth-project-domain-id>]
                [--os-project-domain-name <auth-project-domain-name>]
                [--os-auth-token <auth-token>] [--endpoint <barbican-url>]
                [--interface <barbican-interface>]
                [--service-type <barbican-service-type>]
                [--service-name <barbican-service-name>]
                [--region-name <barbican-region-name>]
                [--barbican-api-version <barbican-api-version>] [--insecure]
                [--os-cacert <ca-certificate>] [--os-cert <certificate>]
                [--os-key <key>] [--timeout <seconds>]
```

2.5.2 barbican optional arguments

--version show programs version number and exit

-v, --verbose Increase verbosity of output. Can be repeated.

-q, --quiet Suppress output except warnings and errors.

--log-file LOG_FILE Specify a file to log output. Disabled by default.

-h, --help Show help message and exit.

--debug Show tracebacks on errors.

--no-auth, -N Do not use authentication.

--os-identity-api-version <identity-api-version> Specify Identity API version to use. Defaults to env[OS_IDENTITY_API_VERSION] or 3.

--os-auth-url <auth-url>, -A <auth-url> Defaults to env[OS_AUTH_URL].

--os-username <auth-user-name>, -U <auth-user-name> Defaults to env[OS_USERNAME].

--os-user-id <auth-user-id> Defaults to env[OS_USER_ID].

--os-password <auth-password>, -P <auth-password> Defaults to env[OS_PASSWORD].

--os-user-domain-id <auth-user-domain-id> Defaults to env[OS_USER_DOMAIN_ID].

--os-user-domain-name <auth-user-domain-name> Defaults to env[OS_USER_DOMAIN_NAME].

--os-tenant-name <auth-tenant-name>, -T <auth-tenant-name> Defaults to env[OS_TENANT_NAME].

--os-tenant-id <tenant-id>, -I <tenant-id> Defaults to env[OS_TENANT_ID].

--os-project-id <auth-project-id> Another way to specify tenant ID. This option is mutually exclusive with os-tenant-id. Defaults to env[OS_PROJECT_ID].

--os-project-name <auth-project-name> Another way to specify tenant name. This option is mutually exclusive with os-tenant-name. Defaults to env[OS_PROJECT_NAME].

--os-project-domain-id <auth-project-domain-id> Defaults to env[OS_PROJECT_DOMAIN_ID].

--os-project-domain-name <auth-project-domain-name> Defaults to env[OS_PROJECT_DOMAIN_NAME].

--os-auth-token <auth-token> Defaults to env[OS_AUTH_TOKEN].

--endpoint <barbican-url>, -E <barbican-url> Defaults to env[BARBICAN_ENDPOINT].

--interface <barbican-interface> Defaults to env[BARBICAN_INTERFACE].

--service-type <barbican-service-type> Defaults to env[BARBICAN_SERVICE_TYPE].

--service-name <barbican-service-name> Defaults to env[BARBICAN_SERVICE_NAME].

--region-name <barbican-region-name> Defaults to env[BARBICAN_REGION_NAME].

--barbican-api-version <barbican-api-version> Defaults to env[BARBICAN_API_VERSION].

barbican acl delete

```
usage: barbican acl delete [-h] URI
```

Delete ACLs for a secret or container as identified by its href.

Positional arguments:

URI The URI reference for the secret or container.

Optional arguments:

-h, --help show this help message and exit

barbican acl get

```
usage: barbican acl get [-h] [-f {csv,html,json,table,value,yaml}] [-c COLUMN]
                        [--max-width <integer>] [--print-empty] [--noindent]
                        [--quote {all,minimal,none,nonnumeric}]
                        URI
```

Retrieve ACLs for a secret or container by providing its href.

Positional arguments:

URI The URI reference for the secret or container.

Optional arguments:

-h, --help show this help message and exit

barbican acl submit

```
usage: barbican acl submit [-h] [-f {csv,html,json,table,value,yaml}]
                           [-c COLUMN] [--max-width <integer>] [--print-empty]
                           [--noindent]
                           [--quote {all,minimal,none,nonnumeric}]
                           [--user [USERS]]
                           [--project-access | --no-project-access]
                           [--operation-type {read}]
                           URI
```

Submit ACL on a secret or container as identified by its href.

Positional arguments:

URI The URI reference for the secret or container.

Optional arguments:

-h, --help show this help message and exit

--user [USERS], -u [USERS] Keystone userid(s) for ACL.

--project-access Flag to enable project access behavior.

--no-project-access Flag to disable project access behavior.

--operation-type {read}, -o {read} Type of Barbican operation ACL is set for

barbican acl user add

```
usage: barbican acl user add [-h] [-f {csv,html,json,table,value,yaml}]
                             [-c COLUMN] [--max-width <integer>]
                             [--print-empty] [--noindent]
                             [--quote {all,minimal,none,nonnumeric}]
                             [--user [USERS]]
                             [--project-access | --no-project-access]
                             [--operation-type {read}]
                             URI
```

Add ACL users to a secret or container as identified by its href.

Positional arguments:

URI The URI reference for the secret or container.

Optional arguments:

-h, --help show this help message and exit

--user [USERS], -u [USERS] Keystone userid(s) for ACL.

--project-access Flag to enable project access behavior.

--no-project-access Flag to disable project access behavior.

--operation-type {read}, -o {read} Type of Barbican operation ACL is set for

barbican acl user remove

```
usage: barbican acl user remove [-h] [-f {csv,html,json,table,value,yaml}]
                                [-c COLUMN] [--max-width <integer>]
                                [--print-empty] [--noindent]
                                [--quote {all,minimal,none,nonnumeric}]
                                [--user [USERS]]
                                [--project-access | --no-project-access]
                                [--operation-type {read}]
                                URI
```

Remove ACL users from a secret or container as identified by its href.

Positional arguments:

URI The URI reference for the secret or container.

Optional arguments:

-h, --help show this help message and exit

--user [USERS], -u [USERS] Keystone userid(s) for ACL.

--project-access Flag to enable project access behavior.

--no-project-access Flag to disable project access behavior.

--operation-type {read}, -o {read} Type of Barbican operation ACL is set for

barbican ca get

```
usage: barbican ca get [-h] [-f {html,json,shell,table,value,yaml}]
                       [-c COLUMN] [--max-width <integer>] [--print-empty]
                       [--noindent] [--prefix PREFIX]
                       URI
```

Retrieve a CA by providing its URI.

Positional arguments:

URI The URI reference for the CA.

Optional arguments:

-h, --help show this help message and exit

barbican ca list

```
usage: barbican ca list [-h] [-f {csv,html,json,table,value,yaml}] [-c COLUMN]
                        [--max-width <integer>] [--print-empty] [--noindent]
                        [--quote {all,minimal,none,nonnumeric}]
                        [--limit LIMIT] [--offset OFFSET] [--name NAME]
```

List CAs.

Optional arguments:

-h, --help show this help message and exit

--limit LIMIT, -l LIMIT specify the limit to the number of items to list per page (default: 10; maximum: 100)

--offset OFFSET, -o OFFSET specify the page offset (default: 0)

--name NAME, -n NAME specify the ca name (default: None)

barbican secret container create

```
usage: barbican secret container create [-h]
                                        [-f {html,json,shell,table,value,yaml}
↪]
                                        [-c COLUMN] [--max-width <integer>]
                                        [--print-empty] [--noindent]
                                        [--prefix PREFIX] [--name NAME]
                                        [--type TYPE] [--secret SECRET]
```

Store a container in Barbican.

Optional arguments:

-h, --help show this help message and exit

--name NAME, -n NAME a human-friendly name.

--type TYPE type of container to create (default: generic).

--secret SECRET, -s SECRET one secret to store in a container (can be set multiple times). Example: secret private_key=https://url.test/v1/secrets/1-2-3-4

barbican secret container delete

```
usage: barbican secret container delete [-h] URI
```

Delete a container by providing its href.

Positional arguments:

URI The URI reference for the container

Optional arguments:

-h, --help show this help message and exit

barbican secret container get

```
usage: barbican secret container get [-h]
                                     [-f {html,json,shell,table,value,yaml}]
                                     [-c COLUMN] [--max-width <integer>]
                                     [--print-empty] [--noindent]
                                     [--prefix PREFIX]
                                     URI
```

Retrieve a container by providing its URI.

Positional arguments:

URI The URI reference for the container.

Optional arguments:

-h, --help show this help message and exit

barbican secret container list

```
usage: barbican secret container list [-h]
                                      [-f {csv,html,json,table,value,yaml}]
                                      [-c COLUMN] [--max-width <integer>]
                                      [--print-empty] [--noindent]
                                      [--quote {all,minimal,none,nonnumeric}]
                                      [--limit LIMIT] [--offset OFFSET]
                                      [--name NAME] [--type TYPE]
```

List containers.

Optional arguments:

-h, --help show this help message and exit

--limit LIMIT, -l LIMIT specify the limit to the number of items to list per page (default: 10; maximum: 100)

--offset OFFSET, -o OFFSET specify the page offset (default: 0)

--name NAME, -n NAME specify the container name (default: None)

--type TYPE, -t TYPE specify the type filter for the list (default: None).

barbican secret delete

```
usage: barbican secret delete [-h] URI
```

Delete a secret by providing its URI.

Positional arguments:

URI The URI reference for the secret

Optional arguments:

-h, --help show this help message and exit

barbican secret get

```
usage: barbican secret get [-h] [-f {html,json,shell,table,value,yaml}]
                           [-c COLUMN] [--max-width <integer>] [--print-empty]
                           [--noindent] [--prefix PREFIX] [--decrypt]
                           [--payload]
                           [--payload_content_type PAYLOAD_CONTENT_TYPE]
                           URI
```

Retrieve a secret by providing its URI.

Positional arguments:

URI The URI reference for the secret.

Optional arguments:

-h, --help show this help message and exit

--decrypt, -d if specified, retrieve the unencrypted secret data; the data type can be specified with payload_content_type.

--payload, -p if specified, retrieve the unencrypted secret data; the data type can be specified with payload_content_type. If the user wishes to only retrieve the value of the payload they must add -f value to format returning only the value of the payload

--payload_content_type PAYLOAD_CONTENT_TYPE, -t PAYLOAD_CONTENT_TYPE the content type of the decrypted secret (default: text/plain).

barbican secret list

```
usage: barbican secret list [-h] [-f {csv,html,json,table,value,yaml}]
                            [-c COLUMN] [--max-width <integer>]
                            [--print-empty] [--noindent]
                            [--quote {all,minimal,none,nonnumeric}]
                            [--limit LIMIT] [--offset OFFSET] [--name NAME]
                            [--algorithm ALGORITHM] [--bit-length BIT_LENGTH]
                            [--mode MODE]
```

List secrets.

Optional arguments:

-h, --help show this help message and exit

--limit LIMIT, -l LIMIT specify the limit to the number of items to list per page (default: 10; maximum: 100)

--offset OFFSET, -o OFFSET specify the page offset (default: 0)

--name NAME, -n NAME specify the secret name (default: None)

--algorithm ALGORITHM, -a ALGORITHM the algorithm filter for the list(default: None).

--bit-length BIT_LENGTH, -b BIT_LENGTH the bit length filter for the list (default: 0).

--mode MODE, -m MODE the algorithm mode filter for the list (default: None).

barbican secret order create

```
usage: barbican secret order create [-h]
                                    [-f {html,json,shell,table,value,yaml}]
                                    [-c COLUMN] [--max-width <integer>]
                                    [--print-empty] [--noindent]
                                    [--prefix PREFIX] [--name NAME]
                                    [--algorithm ALGORITHM]
                                    [--bit-length BIT_LENGTH] [--mode MODE]
                                    [--payload-content-type PAYLOAD_CONTENT_
↪TYPE]
                                    [--expiration EXPIRATION]
                                    [--request-type REQUEST_TYPE]
                                    [--subject-dn SUBJECT_DN]
                                    [--source-container-ref SOURCE_CONTAINER_
↪REF]
                                    [--ca-id CA_ID] [--profile PROFILE]
                                    [--request-file REQUEST_FILE]
                                    type
```

Create a new order.

Positional arguments:

type the type of the order (key, asymmetric, certificate) to create.

Optional arguments:

-h, --help show this help message and exit

--name NAME, -n NAME a human-friendly name.

--algorithm ALGORITHM, -a ALGORITHM the algorithm to be used with the requested key (default: aes).

--bit-length BIT_LENGTH, -b BIT_LENGTH the bit length of the requested secret key (default: 256).

--mode MODE, -m MODE the algorithm mode to be used with the requested key (default: cbc).

--payload-content-type PAYLOAD_CONTENT_TYPE, -t PAYLOAD_CONTENT_TYPE the type/format of the secret to be generated (default: application/octet-stream).

--expiration EXPIRATION, -x EXPIRATION the expiration time for the secret in ISO 8601 format.

--request-type REQUEST_TYPE the type of the certificate request.

--subject-dn SUBJECT_DN the subject of the certificate.

--source-container-ref SOURCE_CONTAINER_REF the source of the certificate when using stored-key requests.

--ca-id CA_ID the identifier of the CA to use for the certificate request.

--profile PROFILE the profile of certificate to use.

--request-file REQUEST_FILE the file containing the CSR.

barbican secret order delete

```
usage: barbican secret order delete [-h] URI
```

Delete an order by providing its href.

Positional arguments:

URI The URI reference for the order

Optional arguments:

-h, --help show this help message and exit

barbican secret order get

```
usage: barbican secret order get [-h] [-f {html,json,shell,table,value,yaml}]
                                 [-c COLUMN] [--max-width <integer>]
                                 [--print-empty] [--noindent]
                                 [--prefix PREFIX]
                                 URI
```

Retrieve an order by providing its URI.

Positional arguments:

URI The URI reference order.

Optional arguments:

-h, --help show this help message and exit

barbican secret order list

```
usage: barbican secret order list [-h] [-f {csv,html,json,table,value,yaml}]
                                  [-c COLUMN] [--max-width <integer>]
                                  [--print-empty] [--noindent]
                                  [--quote {all,minimal,none,nonnumeric}]
                                  [--limit LIMIT] [--offset OFFSET]
```

List orders.

Optional arguments:

-h, --help show this help message and exit

--limit LIMIT, -l LIMIT specify the limit to the number of items to list per page (default: 10; maximum: 100)

--offset OFFSET, -o OFFSET specify the page offset (default: 0)

barbican secret store

```
usage: barbican secret store [-h] [-f {html,json,shell,table,value,yaml}]
                             [-c COLUMN] [--max-width <integer>]
                             [--print-empty] [--noindent] [--prefix PREFIX]
                             [--name NAME] [--payload PAYLOAD]
                             [--secret-type SECRET_TYPE]
                             [--payload-content-type PAYLOAD_CONTENT_TYPE]
                             [--payload-content-encoding PAYLOAD_CONTENT_
↪ENCODING]
                             [--algorithm ALGORITHM] [--bit-length BIT_LENGTH]
                             [--mode MODE] [--expiration EXPIRATION]
```

Store a secret in Barbican.

Optional arguments:

-h, --help show this help message and exit

--name NAME, -n NAME a human-friendly name.

--payload PAYLOAD, -p PAYLOAD the unencrypted secret; if provided, you must also provide a payload_content_type

--secret-type SECRET_TYPE, -s SECRET_TYPE the secret type; must be one of symmetric, public, private, certificate, passphrase, opaque (default)

--payload-content-type PAYLOAD_CONTENT_TYPE, -t PAYLOAD_CONTENT_TYPE the type/format of the provided secret data; text/plain is assumed to be UTF-8; required when payload is supplied.

--payload-content-encoding PAYLOAD_CONTENT_ENCODING, -e PAYLOAD_CONTENT_ENCODING required if payload-content-type is application /octet-stream.

--algorithm ALGORITHM, -a ALGORITHM the algorithm (default: aes).

--bit-length BIT_LENGTH, -b BIT_LENGTH the bit length (default: 256).

--mode MODE, -m MODE the algorithm mode; used only for reference (default: cbc)

--expiration EXPIRATION, -x EXPIRATION the expiration time for the secret in ISO 8601 format.

barbican secret update

```
usage: barbican secret update [-h] URI payload
```

Update a secret with no payload in Barbican.

Positional arguments:

URI The URI reference for the secret.

payload the unencrypted secret

Optional arguments:

-h, --help show this help message and exit

CHAPTER

THREE

CONTRIBUTOR DOCUMENTATION

3.1 Contributing

If you would like to contribute to the development of OpenStack, you must follow the steps in this page:

https://docs.openstack.org/infra/manual/developers.html

Once those steps have been completed, changes to OpenStack should be submitted for review via the Gerrit tool, following the workflow documented at:

https://docs.openstack.org/infra/manual/developers.html#development-workflow

Pull requests submitted through GitHub will be ignored.

Bugs should be filed on OpenStack StoryBoard, not GitHub:

https://storyboard.openstack.org/#!/project/984

3.2 Writing and Running Barbican Client Tests

As a part of every code review that is submitted to the python-barbicanclient project there are a number of gating jobs which aid in the prevention of regression issues within python-barbicanclient. As a result, a python-barbicanclient developer should be familiar with running python-barbicanclient tests locally.

For your convenience we provide the ability to run all tests through the `tox` utility. If you are unfamiliar with tox please see refer to the tox documentation for assistance.

3.2.1 Unit Tests

Currently, we provide tox environments for Python 2.7. By default all available test environments within the tox configuration will execute when calling `tox`. If you want to run them independently, you can do so with the following command:

```
# Executes tests on Python 3.7
tox -e py37
```

Note: If you do not have the appropriate Python versions available, consider setting up PyEnv to install multiple versions of Python. See the documentation setting up a Barbican development environment.

Note: Individual unit tests can also be run, using the following commands:

```
# runs a single test with the function named
# test_should_entity_str
tox -e py37 -- test_should_entity_str

# runs only tests in the WhenTestingSecrets class and
# the WhenTestingOrderManager class
tox -e p37 -- '(WhenTestingSecrets|WhenTestingOrderManager)'
```

The function name or class specified must be one located in the *barbicanclient/tests* directory.

Groups of tests can also be run with a regex match after the `--`. For more information on what can be done with `testr`, please see: http://testrepository.readthedocs.org/en/latest/MANUAL.html

You can also setup breakpoints in the unit tests. This can be done by adding `import pdb; pdb.set_trace()` to the line of the unit test you want to examine, then running the following command:

```
# Executes tests on Python 2.7
tox -e debug
```

Note: For a list of pdb commands, please see: https://docs.python.org/2/library/pdb.html

3.2.2 Functional Tests

Unlike running unit tests, the functional tests require Barbican and Keystone services to be running in order to execute. For more information on setting up a Barbican development environment and using Keystone with Barbican, see our accompanying project documentation.

A configuration file for functional tests must be edited before the tests can be run. In the top-level directory of the python-barbicanclient, edit `/etc/functional_tests.conf` to the values you setup in Keystone.

```
[DEFAULT]
# Leaving this as a placeholder

[keymanager]
# Replace values that represent barbican server and user information
url=http://localhost:9311
username=barbican
password=secretservice
project_name=service
project_id=service
#max_payload_size=10000
project_domain_name=Default

[identity]
# Replace these with values that represent your identity configuration
uri=http://localhost:5000/v2.0
```

(continues on next page)

(continued from previous page)

```
uri_v3=http://localhost:5000/v3
auth_version=v3

username=admin
tenant_name=admin
password=password
domain_name=Default

admin_username=admin
admin_tenant_name=admin
admin_password=password
admin_domain_name=Default


[identity-feature-enabled]
# Leaving this as a placeholder
```

Once you have the appropriate services running and configured you can execute the functional tests through tox.

```
# Execute Barbican Functional Tests
tox -e functional
```

By default, the functional tox job will use nosetests to execute the functional tests. This is primarily due to nose being a very well known and common workflow among developers.

Note: In order to run individual functional test functions, you must use the following commands:

```
# runs only tests in the test_secrets.py file
tox -e functional -- client/v1/functional/test_secrets.py

# runs only tests in the SecretsTestCase class
tox -e functional -- client/v1/functional/test_secrets.py:\
SecretsTestCase

# runs a single test with the function named
# test_secret_create_defaults_check_content_types
tox -e functional -- client/v1/functional/test_secrets.py:\
SecretsTestCase.test_secret_create_defaults_check_content_types
```

The path specified must be one located in the *functionaltests* directory.

3.2.3 Remote Debugging

In order to be able to hit break-points on API calls, you must use remote debugging. This can be done by adding `import rpdb; rpdb.set_trace()` to the line of the API call you wish to test. For example, adding the breakpoint in `def create` in `barbicanclient.secrets.py` will allow you to hit the breakpoint whenever the `create` function is called.

Note: After performing the `POST` the application will freeze. In order to use `rpdb`, you must open up another terminal and run the following:

```
# enter rpdb using telnet
telnet localhost 4444
```

Once in rpdb, you can use the same commands as pdb, as seen here: https://docs.python.org/2/library/pdb.html

---

CHAPTER

FOUR

REFERENCE

4.1 Client

barbicanclient.client.**Client**(*version=None*, *session=None*, **args*, ***kwargs*)

Barbican client used to interact with barbican service.

Parameters

- **version** The API version to use.
- **session** An instance of keystoneauth1.session.Session that can be either authenticated, or not authenticated. When using a non-authenticated Session, you must provide some additional parameters. When no session is provided it will default to a non-authenticated Session.
- **endpoint** Barbican endpoint url. Required when a session is not given, or when using a non-authenticated session. When using an authenticated session, the client will attempt to get an endpoint from the session.
- **project_id** The project ID used for context in Barbican. Required when a session is not given, or when using a non-authenticated session. When using an authenticated session, the project ID will be provided by the authentication mechanism.
- **verify** When a session is not given, the client will create a non-authenticated session. This parameter is passed to the session that is created. If set to False, it allows barbicanclient to perform insecure TLS (https) requests. The servers certificate will not be verified against any certificate authorities. WARNING: This option should be used with caution.
- **service_type** Used as an endpoint filter when using an authenticated keystone session. Defaults to key-manager.
- **service_name** Used as an endpoint filter when using an authenticated keystone session.
- **interface** Used as an endpoint filter when using an authenticated keystone session. Defaults to public.
- **region_name** Used as an endpoint filter when using an authenticated keystone session.

4.2 Secrets

class barbicanclient.v1.secrets.**SecretManager**(*api*)

Entity Manager for Secret entities

create(*name=None*, *payload=None*, *payload_content_type=None*, *payload_content_encoding=None*, *algorithm=None*, *bit_length=None*, *secret_type=None*, *mode=None*, *expiration=None*)

Factory method for creating new *Secret* objects

Secrets returned by this method have not yet been stored in the Barbican service.

Parameters

- **name** A friendly name for the Secret
- **payload** The unencrypted secret data
- **payload_content_type** DEPRECATED: The format/type of the secret data. Setting this can lead to unexpected results. See Launchpad Bug #1419166.
- **payload_content_encoding** DEPRECATED: The encoding of the secret data. Setting this can lead to unexpected results. See Launchpad Bug #1419166.
- **algorithm** The algorithm associated with this secret key
- **bit_length** The bit length of this secret key
- **mode** The algorithm mode used with this secret key
- **secret_type** The secret type for this secret key
- **expiration** The expiration time of the secret in ISO 8601 format

Returns A new Secret object

Return type *barbicanclient.v1.secrets.Secret*

Raises

- *barbicanclient.exceptions.HTTPAuthError* 401 Responses
- *barbicanclient.exceptions.HTTPClientError* 4xx Responses
- *barbicanclient.exceptions.HTTPServerError* 5xx Responses

delete(*secret_ref*)

Delete a Secret from Barbican

Parameters **secret_ref** Full HATEOAS reference to a Secret, or a UUID

Raises

- *barbicanclient.exceptions.HTTPAuthError* 401 Responses
- *barbicanclient.exceptions.HTTPClientError* 4xx Responses
- *barbicanclient.exceptions.HTTPServerError* 5xx Responses

get(*secret_ref*, *payload_content_type=None*)

Retrieve an existing Secret from Barbican

Parameters

- **secret_ref** (*str*) Full HATEOAS reference to a Secret, or a UUID
- **payload_content_type** (*str*) DEPRECATED: Content type to use for payload decryption. Setting this can lead to unexpected results. See Launchpad Bug #1419166.

Returns Secret object retrieved from Barbican

Return type *barbicanclient.v1.secrets.Secret*

Raises

- *barbicanclient.exceptions.HTTPAuthError* 401 Responses
- *barbicanclient.exceptions.HTTPClientError* 4xx Responses
- *barbicanclient.exceptions.HTTPServerError* 5xx Responses

list(*limit=10*, *offset=0*, *name=None*, *algorithm=None*, *mode=None*, *bits=0*, *secret_type=None*, *created=None*, *updated=None*, *expiration=None*, *sort=None*)

List Secrets for the project

This method uses the limit and offset parameters for paging, and also supports filtering.

The time filters (created, updated, and expiration) are expected to be an ISO 8601 formatted string, which can be prefixed with comparison operators: gt: (greater-than), gte: (greater-than-or-equal), lt: (less-than), or lte: (less-than-or-equal).

Parameters

- **limit** Max number of secrets returned
- **offset** Offset secrets to begin list
- **name** Name filter for the list
- **algorithm** Algorithm filter for the list
- **mode** Mode filter for the list
- **bits** Bits filter for the list
- **secret_type** Secret type filter for the list
- **created** Created time filter for the list, an ISO 8601 format string, optionally prefixed with gt:, gte:, lt:, or lte:
- **updated** Updated time filter for the list, an ISO 8601 format string, optionally prefixed with gt:, gte:, lt:, or lte:
- **expiration** Expiration time filter for the list, an ISO 8601 format string, optionally prefixed with gt:, gte:, lt:, or lte:
- **sort** Determines the sorted order of the returned list, a string of comma-separated sort keys (created, expiration, mode, name, secret_type, status, or updated) with a direction appended (:asc or :desc) to each key

Returns list of Secret objects that satisfy the provided filter criteria.

Return type list

Raises

- *barbicanclient.exceptions.HTTPAuthError* 401 Responses
- *barbicanclient.exceptions.HTTPClientError* 4xx Responses
- *barbicanclient.exceptions.HTTPServerError* 5xx Responses

update(*secret_ref*, *payload=None*)

Update an existing Secret in Barbican

Parameters

- **secret_ref** (*str*) Full HATEOAS reference to a Secret, or a UUID
- **payload** (*str*) New payload to add to secret

Raises

- *barbicanclient.exceptions.HTTPAuthError* 401 Responses
- *barbicanclient.exceptions.HTTPClientError* 4xx Responses
- *barbicanclient.exceptions.HTTPServerError* 5xx Responses

class barbicanclient.v1.secrets.**Secret**(*api*, *name=None*, *expiration=None*, *algorithm=None*, *bit_length=None*, *mode=None*, *payload=None*, *payload_content_type=None*, *payload_content_encoding=None*, *secret_ref=None*, *created=None*, *updated=None*, *content_types=None*, *status=None*, *secret_type=None*, *creator_id=None*)

Secrets managed by Barbican

Secrets represent keys, credentials, and other sensitive data that is stored by the Barbican service.

Secret objects should not be instantiated directly.

You should use the *create* or *get* methods of the barbicanclient.secrets.SecretManager instead.

property acls

Get ACL settings for this secret.

delete()

Deletes the Secret from Barbican

property payload

Lazy-loaded property that holds the unencrypted data

store()

Stores the Secret in Barbican.

New Secret objects are not persisted in Barbican until this method is called.

Raises PayloadException

update()

Updates the secret in Barbican.

4.3 Orders

class barbicanclient.v1.orders.**OrderManager**(*api*)

Entity Manager for Order entitites

create_asymmetric(*name=None*, *algorithm=None*, *bit_length=None*, *pass_phrase=None*, *payload_content_type=None*, *expiration=None*)

Factory method for *AsymmetricOrder* objects

AsymmetricOrder objects returned by this method have not yet been submitted to the Barbican service.

Parameters

- **name** A friendly name for the container to be created
- **algorithm** The algorithm associated with this secret key
- **bit_length** The bit length of this secret key
- **pass_phrase** Optional passphrase
- **payload_content_type** The format/type of the secret data
- **expiration** The expiration time of the secret in ISO 8601 format

Returns AsymmetricOrder

Return type *barbicanclient.v1.orders.AsymmetricOrder*

Raises

- *barbicanclient.exceptions.HTTPAuthError* 401 Responses
- *barbicanclient.exceptions.HTTPClientError* 4xx Responses
- *barbicanclient.exceptions.HTTPServerError* 5xx Responses

create_certificate(*name=None*, *request_type=None*, *subject_dn=None*, *source_container_ref=None*, *ca_id=None*, *profile=None*, *request_data=None*)

Factory method for *CertificateOrder* objects

CertificateOrder objects returned by this method have not yet been submitted to the Barbican service.

Parameters

- **name** A friendly name for the container to be created
- **request_type** The type of the certificate request
- **subject_dn** A subject for the certificate
- **source_container_ref** A container with a public/private key pair to use as source for stored-key requests
- **ca_id** The identifier of the CA to use
- **profile** The profile of certificate to use
- **request_data** The CSR content

Returns CertificateOrder

Return type `barbicanclient.v1.orders.CertificateOrder`

`create_key`(*name=None*, *algorithm=None*, *bit_length=None*, *mode=None*, *payload_content_type=None*, *expiration=None*)

Factory method for *KeyOrder* objects

KeyOrder objects returned by this method have not yet been submitted to the Barbican service.

Parameters

- **name** A friendly name for the secret to be created
- **algorithm** The algorithm associated with this secret key
- **bit_length** The bit length of this secret key
- **mode** The algorithm mode used with this secret key
- **payload_content_type** The format/type of the secret data
- **expiration** The expiration time of the secret in ISO 8601 format

Returns KeyOrder

Return type *barbicanclient.v1.orders.KeyOrder*

Raises

- *barbicanclient.exceptions.HTTPAuthError* 401 Responses
- *barbicanclient.exceptions.HTTPClientError* 4xx Responses
- *barbicanclient.exceptions.HTTPServerError* 5xx Responses

`delete`(*order_ref*)

Delete an Order from Barbican

Parameters **order_ref** Full HATEOAS reference to an Order, or a UUID

`get`(*order_ref*)

Retrieve an existing Order from Barbican

Parameters **order_ref** Full HATEOAS reference to an Order, or a UUID

Returns An instance of the appropriate subtype of Order

Raises

- *barbicanclient.exceptions.HTTPAuthError* 401 Responses
- *barbicanclient.exceptions.HTTPClientError* 4xx Responses
- *barbicanclient.exceptions.HTTPServerError* 5xx Responses

`list`(*limit=10*, *offset=0*)

List Orders for the project

This method uses the limit and offset parameters for paging.

Parameters

- **limit** Max number of orders returned

- **offset** Offset orders to begin list

Returns list of Order objects

Raises

- *barbicanclient.exceptions.HTTPAuthError* 401 Responses
- *barbicanclient.exceptions.HTTPClientError* 4xx Responses
- *barbicanclient.exceptions.HTTPServerError* 5xx Responses

class barbicanclient.v1.orders.**Order**(*api*, *type*, *status=None*, *created=None*, *updated=None*, *meta=None*, *order_ref=None*, *error_status_code=None*, *error_reason=None*, *sub_status=None*, *sub_status_message=None*, *creator_id=None*)

Base order object to hold common functionality

This should be considered an abstract class that should not be instantiated directly.

delete()

Deletes the Order from Barbican

submit()

Submit the Order to Barbican.

New Order objects are not persisted in Barbican until this method is called.

class barbicanclient.v1.orders.**KeyOrder**(*api*, *name=None*, *algorithm=None*, *bit_length=None*, *mode=None*, *expiration=None*, *payload_content_type=None*, *status=None*, *created=None*, *updated=None*, *order_ref=None*, *secret_ref=None*, *error_status_code=None*, *error_reason=None*, *sub_status=None*, *sub_status_message=None*, *creator_id=None*)

KeyOrders can be used to request random key material from Barbican

property mode

Encryption mode being used with this key

The mode could be set to CBC for example, when requesting a key that will be used for AES encryption in CBC mode.

class barbicanclient.v1.orders.**AsymmetricOrder**(*api*, *name=None*, *algorithm=None*, *bit_length=None*, *mode=None*, *passphrase=None*, *pass_phrase=None*, *expiration=None*, *payload_content_type=None*, *status=None*, *created=None*, *updated=None*, *order_ref=None*, *container_ref=None*, *error_status_code=None*, *error_reason=None*, *sub_status=None*, *sub_status_message=None*, *creator_id=None*)

property pass_phrase

Passphrase to be used for passphrase protected asymmetric keys

4.4 Containers

class barbicanclient.v1.containers.**ContainerManager**(*api*)

EntityManager for Container entities

You should use the ContainerManager exposed by the Client and should not need to instantiate your own.

create(*name=None*, *secrets=None*)

Factory method for *Container* objects

Container objects returned by this method have not yet been stored in Barbican.

Parameters

- **name** A friendly name for the Container
- **secrets** Secrets to populate when creating a Container

Returns Container

Return type *barbicanclient.v1.containers.Container*

Raises

- *barbicanclient.exceptions.HTTPAuthError* 401 Responses
- *barbicanclient.exceptions.HTTPClientError* 4xx Responses
- *barbicanclient.exceptions.HTTPServerError* 5xx Responses

create_certificate(*name=None*, *certificate=None*, *intermediates=None*, *private_key=None*, *private_key_passphrase=None*)

Factory method for *CertificateContainer* objects

CertificateContainer objects returned by this method have not yet been stored in Barbican.

Parameters

- **name** A friendly name for the CertificateContainer
- **certificate** Secret object containing a Certificate
- **intermediates** Secret object containing Intermediate Certs
- **private_key** Secret object containing a Private Key
- **private_key_passphrase** Secret object containing a passphrase

Returns CertificateContainer

Return type *barbicanclient.v1.containers.CertificateContainer*

Raises

- *barbicanclient.exceptions.HTTPAuthError* 401 Responses
- *barbicanclient.exceptions.HTTPClientError* 4xx Responses

- *barbicanclient.exceptions.HTTPServerError* 5xx Responses

create_rsa(*name=None*, *public_key=None*, *private_key=None*, *private_key_passphrase=None*)

Factory method for *RSAContainer* objects

RSAContainer objects returned by this method have not yet been stored in Barbican.

Parameters

- **name** A friendly name for the RSAContainer
- **public_key** Secret object containing a Public Key
- **private_key** Secret object containing a Private Key
- **private_key_passphrase** Secret object containing a passphrase

Returns RSAContainer

Return type *barbicanclient.v1.containers.RSAContainer*

Raises

- *barbicanclient.exceptions.HTTPAuthError* 401 Responses
- *barbicanclient.exceptions.HTTPClientError* 4xx Responses
- *barbicanclient.exceptions.HTTPServerError* 5xx Responses

delete(*container_ref*)

Delete a Container from Barbican

Parameters **container_ref** Full HATEOAS reference to a Container, or a UUID

Raises

- *barbicanclient.exceptions.HTTPAuthError* 401 Responses
- *barbicanclient.exceptions.HTTPClientError* 4xx Responses
- *barbicanclient.exceptions.HTTPServerError* 5xx Responses

get(*container_ref*)

Retrieve an existing Container from Barbican

Parameters **container_ref** Full HATEOAS reference to a Container, or a UUID

Returns Container object or a subclass of the appropriate type

list(*limit=10*, *offset=0*, *name=None*, *type=None*)

List containers for the project.

This method uses the limit and offset parameters for paging.

Parameters

- **limit** Max number of containers returned
- **offset** Offset containers to begin list
- **name** Name filter for the list

- **type** Type filter for the list

Returns list of Container metadata objects

Raises

- *barbicanclient.exceptions.HTTPAuthError* 401 Responses
- *barbicanclient.exceptions.HTTPClientError* 4xx Responses
- *barbicanclient.exceptions.HTTPServerError* 5xx Responses

register_consumer(*container_ref*, *name*, *url*)

Add a consumer to the container

Parameters

- **container_ref** Full HATEOAS reference to a Container, or a UUID
- **name** Name of the consuming service
- **url** URL of the consuming resource

Returns A container object per the get() method

Raises

- *barbicanclient.exceptions.HTTPAuthError* 401 Responses
- *barbicanclient.exceptions.HTTPClientError* 4xx Responses
- *barbicanclient.exceptions.HTTPServerError* 5xx Responses

remove_consumer(*container_ref*, *name*, *url*)

Remove a consumer from the container

Parameters

- **container_ref** Full HATEOAS reference to a Container, or a UUID
- **name** Name of the previously consuming service
- **url** URL of the previously consuming resource

Raises

- *barbicanclient.exceptions.HTTPAuthError* 401 Responses
- *barbicanclient.exceptions.HTTPClientError* 4xx Responses
- *barbicanclient.exceptions.HTTPServerError* 5xx Responses

class barbicanclient.v1.containers.**Container**(*api*, *name=None*, *secrets=None*, *consumers=None*, *container_ref=None*, *created=None*, *updated=None*, *status=None*, *secret_refs=None*)

Container is a generic grouping of Secrets

property acls

Get ACL settings for this container.

delete()

Delete container from Barbican

property secrets

List of Secrets in Containers

store()

Store Container in Barbican

class barbicanclient.v1.containers.**RSAContainer**(*api*, *name=None*, *public_key=None*, *private_key=None*, *private_key_passphrase=None*, *consumers=[]*, *container_ref=None*, *created=None*, *updated=None*, *status=None*, *public_key_ref=None*, *private_key_ref=None*, *private_key_passphrase_ref=None*)

property private_key

Secret containing the Private Key

property private_key_passphrase

Secret containing the Passphrase

property public_key

Secret containing the Public Key

class barbicanclient.v1.containers.**CertificateContainer**(*api*, *name=None*, *certificate=None*, *intermediates=None*, *private_key=None*, *private_key_passphrase=None*, *consumers=[]*, *container_ref=None*, *created=None*, *updated=None*, *status=None*, *certificate_ref=None*, *intermediates_ref=None*, *private_key_ref=None*, *private_key_passphrase_ref=None*)

property certificate

Secret containing the certificate

property intermediates

Secret containing intermediate certificates

property private_key

Secret containing the private key

property private_key_passphrase

Secret containing the passphrase

4.5 Certificate Authorities

class barbicanclient.v1.cas.**CAManager**(*api*)

Entity Manager for Secret entities

get(*ca_ref*)

Retrieve an existing CA from Barbican

Parameters **ca_ref** (*str*) Full HATEOAS reference to a CA

Returns CA object retrieved from Barbican

Return type *barbicanclient.v1.cas.CA*

Raises

- *barbicanclient.exceptions.HTTPAuthError* 401 Responses
- *barbicanclient.exceptions.HTTPClientError* 4xx Responses
- *barbicanclient.exceptions.HTTPServerError* 5xx Responses

list(*limit=10*, *offset=0*, *name=None*)

List CAs for the project

This method uses the limit and offset parameters for paging, and also supports filtering.

Parameters

- **limit** Max number of CAs returned
- **offset** Offset secrets to begin list
- **name** Name filter for the list

Returns list of CA objects that satisfy the provided filter criteria.

Return type list

Raises

- *barbicanclient.exceptions.HTTPAuthError* 401 Responses
- *barbicanclient.exceptions.HTTPClientError* 4xx Responses
- *barbicanclient.exceptions.HTTPServerError* 5xx Responses

class barbicanclient.v1.cas.**CA**(*api*, *meta=None*, *expiration=None*, *plugin_name=None*, *plugin_ca_id=None*, *ca_ref=None*, *created=None*, *updated=None*, *status=None*, *creator_id=None*)

Certificate authority

CAs represent certificate authorities or subCAs with which the Barbican service is configured to interact.

Certificate authority

CA objects should not be instantiated directly. You should use the *create* or *get* methods of the barbicanclient.cas.CAManager instead.

4.6 ACLs

class barbicanclient.v1.acls.**ACLManager**(*api*)

Entity Manager for Secret or Container ACL entities

create(*entity_ref=None*, *users=None*, *project_access=None*, *operation_type='read'*)

Factory method for creating *ACL* entity.

ACL object returned by this method have not yet been stored in Barbican.

Input entity_ref is used to determine whether ACL object type needs to be `barbicanclient.acls.SecretACL` or `barbicanclient.acls.ContainerACL`.

Parameters

- **entity_ref** (*str*) Full HATEOAS reference to a secret or container
- **users** (*List or None*) List of Keystone userid(s) to be used in ACL.
- **project_access** (*bool*) Flag indicating project access behavior
- **operation_type** (*str*) Type indicating which class of Barbican operations this ACL is defined for e.g. read operations

Returns ACL object instance

Return type *barbicanclient.v1.acls.SecretACL* or *barbicanclient.v1.acls.ContainerACL*

get(*entity_ref*)

Retrieve existing ACLs for a secret or container found in Barbican

Parameters **entity_ref** (*str*) Full HATEOAS reference to a secret or container.

Returns ACL entity object instance

Return type *barbicanclient.v1.acls.SecretACL* or *barbicanclient.v1.acls.ContainerACL*

Raises

- *barbicanclient.exceptions.HTTPAuthError* 401 Responses
- *barbicanclient.exceptions.HTTPClientError* 4xx Responses

class barbicanclient.v1.acls.**SecretACL**(*api*, *entity_ref*, *users=None*, *project_access=None*, *operation_type='read'*, *created=None*, *updated=None*)

ACL entity for a secret

Base ACL entity instance for secret or container.

Provide ACL data arguments to set ACL setting for given operation_type.

To add ACL setting for other operation types, use *add_operation_acl* method.

Parameters

- **api** client instance reference
- **entity_ref** (*str*) Full HATEOAS reference to a secret or container

- **users** (*str List or None*) List of Keystone userid(s) to be used for ACL.
- **project_access** (*bool*) Flag indicating project access behavior
- **operation_type** (*str*) Type indicating which class of Barbican operations this ACL is defined for e.g. read operations
- **created** (*str*) Time string indicating ACL create timestamp. This is populated only when populating data from api response. Not needed in client input.
- **updated** (*str*) Time string indicating ACL last update timestamp. This is populated only when populating data from api response. Not needed in client input.

add_operation_acl(*users=None*, *project_access=None*, *operation_type=None*, *created=None*, *updated=None*)

Add ACL settings to entity for specific operation type.

If matching operation_type ACL already exists, then it replaces it with new PerOperationACL object using provided inputs. Otherwise it appends new PerOperationACL object to existing per operation ACL list.

This just adds to local entity and have not yet applied these changes to server.

Parameters

- **users** (*List or None*) List of Keystone userid(s) to be used in ACL.
- **project_access** (*bool*) Flag indicating project access behavior
- **operation_type** (*str*) Type indicating which class of Barbican operations this ACL is defined for e.g. read operations
- **created** (*str*) Time string indicating ACL create timestamp. This is populated only when populating data from api response. Not needed in client input.
- **updated** (*str*) Time string indicating ACL last update timestamp. This is populated only when populating data from api response. Not needed in client input.

property entity_ref

Entity URI reference.

property entity_uuid

Entity UUID

get(*operation_type*)

Get operation specific ACL instance.

Parameters **operation_type** (*str*) Type indicating which operations ACL setting is needed.

load_acls_data()

Loads ACL entity from Barbican server using its acl_ref

Clears the existing list of per operation ACL settings if there. Populates current ACL entity with ACL settings received from Barbican server.

Raises

- *barbicanclient.exceptions.HTTPAuthError* 401 Responses
- *barbicanclient.exceptions.HTTPClientError* 4xx Responses
- *barbicanclient.exceptions.HTTPServerError* 5xx Responses

property operation_acls

List of operation specific ACL settings.

remove()

Remove Barbican ACLs setting defined for a secret or container

Raises

- *barbicanclient.exceptions.HTTPAuthError* 401 Responses
- *barbicanclient.exceptions.HTTPClientError* 4xx Responses

submit()

Submits ACLs for a secret or a container defined in server

In existing ACL case, this overwrites the existing ACL setting with provided inputs. If input users are None or empty list, this will remove existing ACL users if there. If input project_access flag is None, then default project access behavior is enabled.

Returns str acl_ref: Full HATEOAS reference to a secret or container ACL.

Raises

- *barbicanclient.exceptions.HTTPAuthError* 401 Responses
- *barbicanclient.exceptions.HTTPClientError* 4xx Responses
- *barbicanclient.exceptions.HTTPServerError* 5xx Responses

class barbicanclient.v1.acls.**ContainerACL**(*api*, *entity_ref*, *users=None*, *project_access=None*, *operation_type='read'*, *created=None*, *updated=None*)

ACL entity for a container

Base ACL entity instance for secret or container.

Provide ACL data arguments to set ACL setting for given operation_type.

To add ACL setting for other operation types, use *add_operation_acl* method.

Parameters

- **api** client instance reference
- **entity_ref** (*str*) Full HATEOAS reference to a secret or container
- **users** (*str List or None*) List of Keystone userid(s) to be used for ACL.
- **project_access** (*bool*) Flag indicating project access behavior
- **operation_type** (*str*) Type indicating which class of Barbican operations this ACL is defined for e.g. read operations
- **created** (*str*) Time string indicating ACL create timestamp. This is populated only when populating data from api response. Not needed in client input.

- **updated** (*str*) Time string indicating ACL last update timestamp. This is populated only when populating data from api response. Not needed in client input.

add_operation_acl(*users=None*, *project_access=None*, *operation_type=None*, *created=None*, *updated=None*)

Add ACL settings to entity for specific operation type.

If matching operation_type ACL already exists, then it replaces it with new PerOperationACL object using provided inputs. Otherwise it appends new PerOperationACL object to existing per operation ACL list.

This just adds to local entity and have not yet applied these changes to server.

Parameters

- **users** (*List or None*) List of Keystone userid(s) to be used in ACL.
- **project_access** (*bool*) Flag indicating project access behavior
- **operation_type** (*str*) Type indicating which class of Barbican operations this ACL is defined for e.g. read operations
- **created** (*str*) Time string indicating ACL create timestamp. This is populated only when populating data from api response. Not needed in client input.
- **updated** (*str*) Time string indicating ACL last update timestamp. This is populated only when populating data from api response. Not needed in client input.

property entity_ref

Entity URI reference.

property entity_uuid

Entity UUID

get(*operation_type*)

Get operation specific ACL instance.

Parameters **operation_type** (*str*) Type indicating which operations ACL setting is needed.

load_acls_data()

Loads ACL entity from Barbican server using its acl_ref

Clears the existing list of per operation ACL settings if there. Populates current ACL entity with ACL settings received from Barbican server.

Raises

- *barbicanclient.exceptions.HTTPAuthError* 401 Responses
- *barbicanclient.exceptions.HTTPClientError* 4xx Responses
- *barbicanclient.exceptions.HTTPServerError* 5xx Responses

property operation_acls

List of operation specific ACL settings.

remove()

Remove Barbican ACLs setting defined for a secret or container

Raises

- *barbicanclient.exceptions.HTTPAuthError* 401 Responses
- *barbicanclient.exceptions.HTTPClientError* 4xx Responses

submit()

Submits ACLs for a secret or a container defined in server

In existing ACL case, this overwrites the existing ACL setting with provided inputs. If input users are None or empty list, this will remove existing ACL users if there. If input project_access flag is None, then default project access behavior is enabled.

Returns str acl_ref: Full HATEOAS reference to a secret or container ACL.

Raises

- *barbicanclient.exceptions.HTTPAuthError* 401 Responses
- *barbicanclient.exceptions.HTTPClientError* 4xx Responses
- *barbicanclient.exceptions.HTTPServerError* 5xx Responses

4.7 Exceptions

exception barbicanclient.exceptions.**BarbicanException**

exception barbicanclient.exceptions.**HTTPAuthError**(*message*, *status_code=401*)

Raised for 401 Unauthorized responses from the server.

exception barbicanclient.exceptions.**HTTPClientError**(*message*, *status_code=0*)

Raised for 4xx responses from the server.

exception barbicanclient.exceptions.**HTTPError**(*message*, *status_code=0*)

Base exception for HTTP errors.

exception barbicanclient.exceptions.**HTTPServerError**(*message*, *status_code=0*)

Raised for 5xx responses from the server.

exception barbicanclient.exceptions.**PayloadException**

exception barbicanclient.exceptions.**UnsupportedVersion**

User is trying to use an unsupported version of the API.

Indices and tables

- genindex
- modindex
- search

PYTHON MODULE INDEX

b

INDEX

S

U