OpenStackSDK Documentation

Release 4.5.0.dev19

OpenStack Foundation

Mar 10, 2025

CONTENTS

This documentation is split into three sections:

- An *installation* guide
- A section for *users* looking to build applications which make use of OpenStack
- A section for those looking to *contribute* to this project

CHAPTER

ONE

INSTALLATION

1.1 Installation guide

The OpenStack SDK is available on PyPI under the name **openstacksdk**. To install it, use `pip`:

```
$ pip install openstacksdk
```

To check the installed version you can call the module with:

```
$ python -m openstack version
```

CHAPTER

TWO

FOR USERS

2.1 Using the OpenStack SDK

This section of documentation pertains to those who wish to use this SDK in their own application. If youre looking for documentation on how to contribute to or extend the SDK, refer to the contributor section.

For a listing of terms used throughout the SDK, including the names of projects and services supported by it, see the *glossary*.

2.1.1 User Guides

These guides walk you through how to make use of the libraries we provide to work with each OpenStack service. If youre looking for a cookbook approach, this is where youll want to begin.

Getting started

openstacksdk aims to talk to any OpenStack cloud. To do this, it requires a configuration file. openstacksdk favours `clouds.yaml` files, but can also use environment variables. The `clouds.yaml` file should be provided by your cloud provider or deployment tooling. An example:

```
clouds:
 mordred:
   region_name: Dallas
   auth:
     username: 'mordred'
     password: XXXXXXX
     project_name: 'demo'
     auth_url: 'https://identity.example.com'
```

More information on configuring openstacksdk can be found in *Configuring OpenStack SDK Applications*.

Given sufficient configuration, you can use openstacksdk to interact with your cloud. openstacksdk consists of three layers. Most users will make use of the *proxy* layer. Using the above `clouds.yaml`, consider listing servers:

```
import openstack

# Initialize and turn on debug logging
openstack.enable_logging(debug=True)
```

(continues on next page)

(continued from previous page)

```
# Initialize connection
conn = openstack.connect(cloud='mordred')

# List the servers
for server in conn.compute.servers():
    print(server.to_dict())
```

openstacksdk also contains a higher-level *cloud* layer based on logical operations:

```
import openstack

# Initialize and turn on debug logging
openstack.enable_logging(debug=True)

# Initialize connection
conn = openstack.connect(cloud='mordred')

# List the servers
for server in conn.list_servers():
    print(server.to_dict())
```

The benefit of this layer is mostly seen in more complicated operations that take multiple steps and where the steps vary across providers. For example:

```
import openstack

# Initialize and turn on debug logging
openstack.enable_logging(debug=True)

# Initialize connection
conn = openstack.connect(cloud='mordred')

# Upload an image to the cloud
image = conn.create_image(
    'ubuntu-trusty', filename='ubuntu-trusty.qcow2', wait=True)

# Find a flavor with at least 512M of RAM
flavor = conn.get_flavor_by_ram(512)

# Boot a server, wait for it to boot, and then do whatever is needed
# to get a public IP address for it.
conn.create_server(
    'my-server', image=image, flavor=flavor, wait=True, auto_ip=True)
```

Finally, there is the low-level *resource* layer. This provides support for the basic CRUD operations supported by REST APIs and is the base building block for the other layers. You typically will not need to use this directly:

```
import openstack
import openstack.config.loader
```

(continues on next page)

(continued from previous page)

```
import openstack.compute.v2.server

# Initialize and turn on debug logging
openstack.enable_logging(debug=True)

# Initialize connection
conn = openstack.connect(cloud='mordred')

# List the servers
for server in openstack.compute.v2.server.Server.list(session=conn.compute):
    print(server.to_dict())
```

Using os-client-config

Configuring OpenStack SDK Applications

Environment Variables

openstacksdk honors all of the normal *OS_** variables. It does not provide backwards compatibility to service-specific variables such as *NOVA_USERNAME*.

If you have OpenStack environment variables set, *openstacksdk* will produce a cloud config object named *envvars* containing your values from the environment. If you dont like the name *envvars*, thats ok, you can override it by setting *OS_CLOUD_NAME*.

Service specific settings, like the nova service type, are set with the default service type as a prefix. For instance, to set a special service_type for trove set

```
export OS_DATABASE_SERVICE_TYPE=rax:database
```

Config Files

openstacksdk will look for a file called *clouds.yaml* in the following locations:

- `.` (the current directory)
- `$HOME/.config/openstack`
- `/etc/openstack`

The first file found wins.

You can also set the environment variable *OS_CLIENT_CONFIG_FILE* to an absolute path of a file to look for and that location will be inserted at the front of the file search list.

The keys are all of the keys youd expect from *OS_** - except lower case and without the OS prefix. So, region name is set with *region_name*.

Service specific settings, like the nova service type, are set with the default service type as a prefix. For instance, to set a special service_type for trove (because youre using Rackspace) set:

```
database_service_type: 'rax:database'
```

Site Specific File Locations

In addition to *~/.config/openstack* and */etc/openstack* - some platforms have other locations they like to put things. *openstacksdk* will also look in an OS specific config dir

- *USER_CONFIG_DIR*
- *SITE_CONFIG_DIR*

USER_CONFIG_DIR is different on Linux, OSX and Windows.

- Linux: *~/.config/openstack*
- OSX: *~/Library/Application Support/openstack*
- Windows: *C:\Users\USERNAME\AppData\Local\OpenStack\openstack*

SITE_CONFIG_DIR is different on Linux, OSX and Windows.

- Linux: */etc/openstack*
- OSX: */Library/Application Support/openstack*
- Windows: *C:\ProgramData\OpenStack\openstack*

An example config file is probably helpful:

```
clouds:
  mtvexx:
    profile: https://vexxhost.com
    auth:
      username: mordred@inaugust.com
      password: XXXXXXXXX
      project_name: mordred@inaugust.com
    region_name: ca-ymq-1
    dns_api_version: 1
  mordred:
    region_name: RegionOne
    auth:
      username: 'mordred'
      password: XXXXXXX
      project_name: 'shade'
      auth_url: 'https://montytaylor-sjc.openstack.blueboxgrid.com:5001/v2.0'
  infra:
    profile: rackspace
    auth:
      username: openstackci
      password: XXXXXXXX
      project_id: 610275
    regions:
    - DFW
    - ORD
    - IAD
```

You may note a few things. First, since *auth_url* settings are silly and embarrassingly ugly, known cloud vendor profile information is included and may be referenced by name or by base URL to the cloud in question if the cloud serves a vendor profile. One of the benefits of that is that *auth_url* isnt the only thing

the vendor defaults contain. For instance, since Rackspace lists *rax:database* as the service type for trove, *openstacksdk* knows that so that you dont have to. In case the cloud vendor profile is not available, you can provide one called *clouds-public.yaml*, following the same location rules previously mentioned for the config files.

regions can be a list of regions. When you call *get_all_clouds*, youll get a cloud config object for each cloud/region combo.

As seen with *dns_service_type*, any setting that makes sense to be per-service, like *service_type* or *endpoint* or *api_version* can be set by prefixing the setting with the default service type. That might strike you funny when setting *service_type* and it does me too - but thats just the world we live in.

Auth Settings

Keystone has auth plugins - which means its not possible to know ahead of time which auth settings are needed. *openstacksdk* sets the default plugin type to *password*, which is what things all were before plugins came about. In order to facilitate validation of values, all of the parameters that exist as a result of a chosen plugin need to go into the auth dict. For password auth, this includes *auth_url*, *username* and *password* as well as anything related to domains, projects and trusts.

API Settings

The following settings are passed to keystoneauth and are common to all services.

api_timeout
: A timeout for API requests. This should be a numerical value indicating some amount (or fraction) of seconds or 0 for no timeout. (optional, defaults to 0)

collect_timing
: Whether or not to collect per-method timing information for each API call. (optional, defaults to False)

Splitting Secrets

In some scenarios, such as configuration management controlled environments, it might be easier to have secrets in one file and non-secrets in another. This is fully supported via an optional file *secure.yaml* which follows all the same location rules as *clouds.yaml*. It can contain anything you put in *clouds.yaml* and will take precedence over anything in the *clouds.yaml* file.

```
# clouds.yaml
clouds:
  internap:
    profile: internap
    auth:
      username: api-55f9a00fb2619
      project_name: inap-17037
    regions:
    - ams01
    - nyj01
# secure.yaml
clouds:
  internap:
```

(continues on next page)

(continued from previous page)

```
    auth:
      password: XXXXXXXXXXXXXXXX
```

SSL Settings

When the access to a cloud is done via a secure connection, *openstacksdk* will always verify the SSL cert by default. This can be disabled by setting *verify* to *False*. In case the cert is signed by an unknown CA, a specific cacert can be provided via *cacert*. **WARNING:** *verify* will always have precedence over *cacert*, so when setting a CA cert but disabling *verify*, the cloud cert will never be validated.

Client certs are also configurable. *cert* will be the client cert file location. In case the cert key is not included within the client cert file, its file location needs to be set via *key*.

```
# clouds.yaml
clouds:
  regular-secure-cloud:
    auth:
      auth_url: https://signed.cert.domain:5000
      ...
  unknown-ca-with-client-cert-secure-cloud:
    auth:
      auth_url: https://unknown.ca.but.secure.domain:5000
      ...
    key: /home/myhome/client-cert.key
    cert: /home/myhome/client-cert.crt
    cacert: /home/myhome/ca.crt
  self-signed-insecure-cloud:
    auth:
      auth_url: https://self.signed.cert.domain:5000
      ...
    verify: False
```

Note for parity with `openstack` command-line options the *insecure* boolean is also recognised (with the opposite semantics to *verify*; i.e. *True* ignores certificate failures). This should be considered deprecated for *verify*.

Cache Settings

Changed in version 1.0.0: Previously, caching was managed exclusively in the cloud layer. Starting in openstacksdk 1.0.0, caching is moved to the proxy layer. As the cloud layer depends on the proxy layer in 1.0.0, this means both layers can benefit from the cache.

Authenticating and accessing resources on a cloud is often expensive. It is therefore quite common that applications will wish to do some client-side caching of both credentials and cloud resources. To facilitate this, *openstacksdk* supports caching credentials and resources using the system keyring and *dogpile.cache*, respectively.

Tip

It is important to emphasise that *openstacksdk* does not actually cache anything itself. Rather, it

collects and presents the cache information so that your various applications that are connecting to OpenStack can share a cache should you desire. It is important that your cache backend is correctly configured according to the needs of your application.

Caching in enabled or disabled globally, rather than on a cloud-by-cloud basis. This is done by setting configuring the“cache“ top-level key. Caching of authentication tokens can be configured using the following settings:

cache.auth
: A boolean indicating whether tokens should be cached in the keyring. When enabled, this allows the consequent connections to the same cloud to skip fetching new token. When the token expires or is invalidated, *openstacksdk* will automatically establish a new connection. Defaults to `false`.

For example, to configure caching of authentication tokens.

```
cache:
  auth: true
```

Caching of resources can be configured using the following settings:

cache.expiration_time
: The expiration time in seconds for a cache entry. This should be an integer. Defaults to `0`.

cache.class
: The cache backend to use, which can include any backend supported by *dogpile.cache* natively as well as backend provided by third-part packages. This should be a string. Defaults to `dogpile.cache.memory`.

cache.arguments
: A mapping of arbitrary arguments to pass into the cache backend. These are backend specific. Keys should correspond to a configuration option for the configured cache backend. Defaults to `{}`.

cache.expirations
: A mapping of resource types to expiration times. The keys should be specified in the same way as the metrics are emitted, by joining meaningful resource URL segments with `.`. For example, both `/servers` and `/servers/ID` should be specified as `servers`, while `/servers/ID/metadata/KEY` should be specified as *server.metadata*. Values should be an expiration time in seconds. A value of `-1` indicates that the cache should never expire, while a value of `0` disables caching for the resource. Defaults to `{}`

For example, to configure caching with the `dogpile.cache.memory` backend with a 1 hour expiration.

```
cache:
  expiration_time: 3600
```

To configure caching with the `dogpile.cache.memory` backend with a 1 hour expiration but only for requests to the OpenStack Compute services `/servers` API:

```
cache:
  expirations:
    servers: 3600
```

To configure caching with the `dogpile.cache.pylibmc` backend with a 1 hour expiration time and a memcached server running on your localhost.

```
cache:
  expiration_time: 3600
  arguments:
    url:
    - 127.0.0.1
```

To configure caching with the `dogpile.cache.pylibmc` backend with a 1 hour expiration time, a memcached server running on your localhost, and multiple per-resource cache expiration times.

```
cache:
  class: dogpile.cache.pylibmc
  expiration_time: 3600
  arguments:
    url:
      - 127.0.0.1
  expiration:
    server: 5
    flavor: -1
    compute.servers: 5
    compute.flavors: -1
    image.images: 5
```

Finally, if the `cache` key is undefined, a null cache is enabled meaning caching is effectively disabled.

Note

Non `GET` requests cause cache invalidation based on the caching key prefix. This means that, for example, a `PUT` request to `/images/ID` will invalidate all images cache (list and all individual entries). Moreover it is possible to explicitly pass the `skip_cache` parameter to the `proxy._get` function to bypass cache and invalidate what is already there. This is happening automatically in the `wait_for_status` methods where it is expected that resource will change some of the attributes over the time. Forcing complete cache invalidation can be achieved calling `conn._cache.invalidate`

MFA Support

MFA support requires a specially prepared configuration file. In this case a combination of two different authorization plugins is used with their individual requirements to the specified parameters.

```
clouds:
  mfa:
    auth_type: "v3multifactor"
    auth_methods:
      - v3password
      - v3totp
    auth:
      auth_url: https://identity.cloud.com
      username: user
      user_id: uid
```

(continues on next page)

(continued from previous page)

```
      password: XXXXXXXXX
      project_name: project
      user_domain_name: udn
      project_domain_name: pdn
```

IPv6

IPv6 is the future, and you should always use it if your cloud supports it and if your local network supports it. Both of those are easily detectable and all friendly software should do the right thing.

However, sometimes a cloud API may return IPv6 information that is not useful to a production deployment. For example, the API may provide an IPv6 address for a server, but not provide that to the host instance via metadata (configdrive) or standard IPv6 autoconfiguration methods (i.e. the host either needs to make a bespoke API call, or otherwise statically configure itself).

For such situations, you can set the `force_ipv4`, or `OS_FORCE_IPV4` boolean environment variable. For example:

```
clouds:
  mtvexx:
    profile: vexxhost
    auth:
      username: mordred@inaugust.com
      password: XXXXXXXXX
      project_name: mordred@inaugust.com
    region_name: ca-ymq-1
    dns_api_version: 1
  monty:
    profile: fooprovider
    force_ipv4: true
    auth:
      username: mordred@inaugust.com
      password: XXXXXXXXX
      project_name: mordred@inaugust.com
    region_name: RegionFoo
```

The above snippet will tell client programs to prefer the IPv4 address and leave the `public_v6` field of the *Server* object blank for the `fooprovider` cloud . You can also set this with a client flag for all clouds:

```
client:
  force_ipv4: true
```

Per-region settings

Sometimes you have a cloud provider that has config that is common to the cloud, but also with some things you might want to express on a per-region basis. For instance, Internap provides a public and private network specific to the user in each region, and putting the values of those networks into config can make consuming programs more efficient.

To support this, the region list can actually be a list of dicts, and any setting that can be set at the cloud level can be overridden for that region.

```yaml
clouds:
  internap:
    profile: internap
    auth:
      password: XXXXXXXXXXXXXXXX
      username: api-55f9a00fb2619
      project_name: inap-17037
    regions:
    - name: ams01
      values:
        networks:
        - name: inap-17037-WAN1654
          routes_externally: true
        - name: inap-17037-LAN6745
    - name: nyj01
      values:
        networks:
        - name: inap-17037-WAN1654
          routes_externally: true
        - name: inap-17037-LAN6745
```

Setting Precedence

Some settings are redundant, e.g. `project-name` and `project-id` both specify the project. In a conflict between redundant settings, the `_name` `clouds.yaml` option (or equivalent `-name` CLI option and `_NAME` environment variable) will be used.

Some environment variables or commandline flags can override the settings from clouds.yaml. These are:

- `--domain-id` (`OS_DOMAIN_ID`)
- `--domain-name` (`OS_DOMAIN_NAME`)
- `--user-domain-id` (`OS_USER_DOMAIN_ID`)
- `--user-domain-name` (`OS_USER_DOMAIN_NAME`)
- `--project-domain-id` (`OS_PROJECT_DOMAIN_ID`)
- `--project-domain-name` (`OS_PROJECT_DOMAIN_NAME`)
- `--auth-token` (`OS_AUTH_TOKEN`)
- `--project-id` (`OS_PROJECT_ID`)
- `--project-name` (`OS_PROJECT_NAME`)
- `--tenant-id` (`OS_TENANT_ID`) (deprecated for `--project-id`)
- `--tenant-name` (`OS_TENANT_NAME`) (deprecated for `--project-name`)

Similarly, if one of the above settings is specified in `clouds.yaml` as part of the `auth` section as well as the main section, the `auth` settings will be overridden. For example in this config section, note that project is specified multiple times:

```
clouds:
  mtvexx:
    profile: https://vexxhost.com
    auth:
      username: mordred@inaugust.com
      password: XXXXXXXXX
      project_name: mylessfavoriteproject
      project_id: 0bedab75-898c-4521-a038-0b4b71c41bed
    region_name: ca-ymq-1
    project_name: myfavoriteproject
    project_id: 2acf9403-25e8-479e-a3c6-d67540c424a4
```

In the above example, the `project_id` configuration values will be ignored in favor of the `project_name` configuration values, and the higher-level project will be chosen over the auth-specified project. So the actual project used will be `` `myfavoriteproject` ``.

Examples

auth

Password-based authentication (project-scoped)

```
example:
    auth:
        auth_url: http://example.com/identity
        password: password
        project_domain_id: default
        project_name: admin
        user_domain_id: default
        username: admin
    region_name: RegionOne
```

Password-based authentication (domain-scoped)

```
example:
    auth:
        auth_url: http://example.com/identity
        domain_id: default
        password: password
        username: admin
    region_name: RegionOne
```

Password-based authentication (trust-scoped)

```
example-trust:
    auth:
        auth_url: http://example.com/identity
        password: password
        username: admin
```

(continues on next page)

(continued from previous page)

```
        trust_id: 95946f9eef864fdc993079d8fe3e5747
    region_name: RegionOne
```

Password-based authentication (system-scoped)

```
example-system:
    auth:
        auth_url: http://example.com/identity
        password: password
        system_scope: all
        username: admin
    region_name: RegionOne
```

Application credential-based authentication

```
example-appcred:
    auth:
        auth_url: http://example.com/identity
        application_credential_id: 9da0a8da3d394d09bf49dfc27014d254
        application_credential_secret: pKfDSvUOFwO2t2_
↪XxCajAFhzCKAVHI7yfqPb6xjshVDnMUHF7ifju8gMdhHTI4Eo56UP_hEc8ssmgA1NNtKMpA
    auth_type: v3applicationcredential
    region_name: RegionOne
```

Token-based authentication

```
example-token:
    auth:
        auth_url: http://example.com/identity
        token:␣
↪gAAAAABl32ptw2PN6L9JyBeO16PwQU1SrdMUvUz8Eon7LC2PFItdGRWFpOkK0qwH3JkukTuEM5qbYK9ucowRXET1RI
↪febVGaW9oJzf6R3WTMDxWz3YRJjmiOBpwcN8
        project_id: 1fd93a4455c74d2ea94b929fc5f0e488
    auth_type: v3token
    region_name: RegionOne
```

Note

This is a toy example: by their very definition tokens are short-lived. You are unlikely to store them in a `clouds.yaml` file. Instead, you would likely pass the TOTP token via the command line (`--os-token`) or as an environment variable (`OS_TOKEN`).

TOTP-based authentication

```
example-totp:
    auth:
        auth_url: http://example.com/identity
```

(continues on next page)

(continued from previous page)

```
        passcode: password
        project_domain_id: default
        project_name: admin
        user_domain_id: default
        username: admin
    auth_type: v3totp
    region_name: RegionOne
```

Note

This is a toy example: by their very definition TOTP tokens are short-lived. You are unlikely to store them in a `clouds.yaml` file. Instead, you would likely pass the TOTP token via the command line (`--os-passcode`) or as an environment variable (`OS_PASSCODE`).

OAuth1-based authentication

```
example-oauth:
    auth:
        auth_url: http://example.com/identity
        consumer_key: foo
        consumer_secret: secret
        access_key: bar
        access_secret: secret
    auth_type: v3oauth1
    region_name: RegionOne
```

Using openstack.config in an Application

Usage

The simplest and least useful thing you can do is:

```
python -m openstack.config.loader
```

Which will print out whatever if finds for your config. If you want to use it from python, which is much more likely what you want to do, things like:

Get a named cloud.

```
import openstack.config

cloud_region = openstack.config.OpenStackConfig().get_one(
    'internap', region_name='ams01')
print(cloud_region.name, cloud_region.region, cloud_region.config)
```

Or, get all of the clouds.

```
import openstack.config
```

(continues on next page)

(continued from previous page)

```
cloud_regions = openstack.config.OpenStackConfig().get_all()
for cloud_region in cloud_regions:
    print(cloud_region.name, cloud_region.region, cloud_region.config)
```

argparse

If youre using *openstack.config* from a program that wants to process command line options, there is a registration function to register the arguments that both *openstack.config* and keystoneauth know how to deal with - as well as a consumption argument.

```
import argparse

import openstack

parser = argparse.ArgumentParser()
cloud = openstack.connect(options=parser)
```

Vendor Support

OpenStack presents deployers with many options, some of which can expose differences to end users. *os-client-config* tries its best to collect information about various things a user would need to know. The following is a text representation of the vendor related defaults *os-client-config* knows about.

Default Values

These are the default behaviors unless a cloud is configured differently.

- Identity uses *password* authentication
- Identity API Version is 2
- Image API Version is 2
- Volume API Version is 2
- Compute API Version is 2.1
- Images must be in *qcow2* format
- Images are uploaded using PUT interface
- Public IPv4 is directly routable via DHCP from Neutron
- IPv6 is not provided
- Floating IPs are not required
- Floating IPs are provided by Neutron
- Security groups are provided by Neutron
- Vendor specific agents are not used

AURO

https://api.auro.io:5000/v2.0

Region Name	Location
van1	Vancouver, BC

- Public IPv4 is provided via NAT with Neutron Floating IP

Betacloud

https://api-1.betacloud.de:5000

Region Name	Location
betacloud-1	Karlsruhe, Germany

- Identity API Version is 3
- Images must be in *raw* format
- Public IPv4 is provided via NAT with Neutron Floating IP
- Volume API Version is 3

Binero

https://auth.binero.cloud:5000/v3

Region Name	Location
europe-se-1	Stockholm, SE

- Identity API Version is 3
- Volume API Version is 3
- Public IPv4 is directly routable via DHCP from Neutron
- Public IPv4 is provided via NAT with Neutron Floating IP

Catalyst

https://api.cloud.catalyst.net.nz:5000/v2.0

Region Name	Location
nz-por-1	Porirua, NZ
nz_wlg_2	Wellington, NZ

- Identity API Version is 3
- Compute API Version is 2
- Images must be in *raw* format

- Volume API Version is 3

City Cloud

https://%(region_name)s.citycloud.com:5000/v3/

Region Name	Location
Buf1	Buffalo, NY
dx1	Dubai, UAE
Fra1	Frankfurt, DE
Kna1	Karlskrona, SE
Lon1	London, UK
Sto2	Stockholm, SE
tky1	Tokyo, JP

- Identity API Version is 3
- Public IPv4 is provided via NAT with Neutron Floating IP
- Volume API Version is 1

ConoHa

https://identity.%(region_name)s.conoha.io

Region Name	Location
tyo1	Tokyo, JP
sin1	Singapore
sjc1	San Jose, CA

- Image upload is not supported

DreamCompute

https://iad2.dream.io:5000

Region Name	Location
RegionOne	Ashburn, VA

- Identity API Version is 3
- Images must be in *raw* format
- IPv6 is provided to every server

Open Telekom Cloud

https://iam.%(region_name)s.otc.t-systems.com/v3

Region Name	Location
eu-de	Biere/Magdeburg, DE
eu-nl	Amsterdam, NL

- Identity API Version is 3
- Public IPv4 is provided via NAT with Neutron Floating IP

ELASTX

https://ops.elastx.cloud:5000/v3

Region Name	Location
se-sto	Stockholm, SE

- Identity API Version is 3
- Public IPv4 is provided via NAT with Neutron Floating IP

Enter Cloud Suite

https://api.entercloudsuite.com/v2.0

Region Name	Location
nl-ams1	Amsterdam, NL
it-mil1	Milan, IT
de-fra1	Frankfurt, DE

- Compute API Version is 2

Fuga

https://identity.api.fuga.io:5000

Region Name	Location
cystack	Netherlands

- Identity API Version is 3
- Volume API Version is 3

Internap

https://identity.api.cloud.inap.com/v2.0

Region Name	Location
ams01	Amsterdam, NL
da01	Dallas, TX
nyj01	New York, NY
sin01	Singapore
sjc01	San Jose, CA

- Floating IPs are not supported

Limestone Networks

https://auth.cloud.lstn.net:5000/v3

Region Name	Location
us-dfw-1	Dallas, TX
us-slc	Salt Lake City, UT

- Identity API Version is 3
- Images must be in *raw* format
- IPv6 is provided to every server connected to the *Public Internet* network

OVH

https://auth.cloud.ovh.net/v3

Region Name	Location
BHS1	Beauharnois, QC
SBG1	Strassbourg, FR
GRA1	Gravelines, FR

- Images may be in *raw* format. The *qcow2* default is also supported
- Floating IPs are not supported

Rackspace

https://identity.api.rackspacecloud.com/v2.0/

Region Name	Location
DFW	Dallas, TX
HKG	Hong Kong
IAD	Washington, D.C.
LON	London, UK
ORD	Chicago, IL
SYD	Sydney, NSW

- Database Service Type is *rax:database*

- Compute Service Name is *cloudServersOpenStack*
- Images must be in *vhd* format
- Images must be uploaded using the Glance Task Interface
- Floating IPs are not supported
- Public IPv4 is directly routable via static config by Nova
- IPv6 is provided to every server
- Security groups are not supported
- Uploaded Images need properties to not use vendor agent:: :vm_mode: hvm :xenapi_use_agent: False
- Block Storage API Version is 2
- The Block Storage API supports version 2 but only version 1 is in the catalog. The Block Storage endpoint is https://{region_name}.blockstorage.api.rackspacecloud.com/v2/{project_id}
- While passwords are recommended for use, API keys do work as well. The *rackspaceauth* python package must be installed, and then the following can be added to clouds.yaml:

```
auth:
  username: myusername
  api_key: myapikey
auth_type: rackspace_apikey
```

SWITCHengines

https://keystone.cloud.switch.ch:5000/v3

Region Name	Location
LS	Lausanne, CH
ZH	Zurich, CH

- Identity API Version is 3
- Compute API Version is 2
- Images must be in *raw* format
- Volume API Version is 3

Ultimum

https://console.ultimum-cloud.com:5000/v2.0

Region Name	Location
RegionOne	Prague, CZ

- Volume API Version is 1

UnitedStack

https://identity.api.ustack.com/v3

Region Name	Location
bj1	Beijing, CN
gd1	Guangdong, CN

- Identity API Version is 3
- Images must be in *raw* format
- Volume API Version is 1

VEXXHOST

http://auth.vexxhost.net

Region Name	Location
ca-ymq-1	Montreal, QC
sjc1	Santa Clara, CA

- DNS API Version is 1
- Identity API Version is 3
- Volume API Version is 3

Zetta

https://identity.api.zetta.io/v3

Region Name	Location
no-osl1	Oslo, NO

- DNS API Version is 2
- Identity API Version is 3

Network Config

There are several different qualities that networks in OpenStack might have that might not be able to be automatically inferred from the available metadata. To help users navigate more complex setups, *os-client-config* allows configuring a list of network metadata.

```
clouds:
  amazing:
    networks:
    - name: blue
      routes_externally: true
    - name: purple
```

(continues on next page)

(continued from previous page)

```
    routes_externally: true
    default_interface: true
  - name: green
    routes_externally: false
  - name: yellow
    routes_externally: false
    nat_destination: true
  - name: chartreuse
    routes_externally: false
    routes_ipv6_externally: true
  - name: aubergine
    routes_ipv4_externally: false
    routes_ipv6_externally: true
```

Every entry must have a name field, which can hold either the name or the id of the network.

routes_externally is a boolean field that labels the network as handling north/south traffic off of the cloud. In a public cloud this might be thought of as the public network, but in private clouds its possible it might be an RFC1918 address. In either case, its provides IPs to servers that things not on the cloud can use. This value defaults to *false*, which indicates only servers on the same network can talk to it.

routes_ipv4_externally and *routes_ipv6_externally* are boolean fields to help handle *routes_externally* in the case where a network has a split stack with different values for IPv4 and IPv6. Either entry, if not given, defaults to the value of *routes_externally*.

default_interface is a boolean field that indicates that the network is the one that programs should use. It defaults to false. An example of needing to use this value is a cloud with two private networks, and where a user is running ansible in one of the servers to talk to other servers on the private network. Because both networks are private, there would otherwise be no way to determine which one should be used for the traffic. There can only be one *default_interface* per cloud.

nat_destination is a boolean field that indicates which network floating ips should be attached to. It defaults to false. Normally this can be inferred by looking for a network that has subnets that have a gateway_ip. But its possible to have more than one network that satisfies that condition, so the user might want to tell programs which one to pick. There can be only one *nat_destination* per cloud.

nat_source is a boolean field that indicates which network floating ips should be requested from. It defaults to false. Normally this can be inferred by looking for a network that is attached to a router. But its possible to have more than one network that satisfies that condition, so the user might want to tell programs which one to pick. There can be only one *nat_source* per cloud.

API Reference

class openstack.config.**OpenStackConfig**(*config_files=None*, *vendor_files=None*, *override_defaults=None*, *force_ipv4=None*, *envvar_prefix=None*, *secure_files=None*, *pw_func=None*, *session_constructor=None*, *app_name=None*, *app_version=None*, *load_yaml_config=True*, *load_envvars=True*, *statsd_host=None*, *statsd_port=None*, *statsd_prefix=None*, *influxdb_config=None*)

get_extra_config(*key*, *defaults=None*)

Fetch an arbitrary extra chunk of config, laying in defaults.

Parameters

- **key** (*string*) name of the config section to fetch
- **defaults** (*dict*) (optional) default values to merge under the found config

register_argparse_arguments(*parser*, *argv*, *service_keys=None*)

Register all of the common argparse options needed.

Given an argparse parser, register the keystoneauth Session arguments, the keystoneauth Auth Plugin Options and os-cloud. Also, peek in the argv to see if all of the auth plugin options should be registered or merely the ones already configured.

Parameters

- **argparse.ArgumentParser** parser to attach argparse options to
- **argv** the arguments provided to the application
- **service_keys** (*string*) Service or list of services this argparse should be specialized for, if known. The first item in the list will be used as the default value for service_type (optional)

:raises exceptions.ConfigException if an invalid auth-type is requested

auth_config_hook(*config*)

Allow examination of config values before loading auth plugin

OpenStackClient will override this to perform additional checks on auth_type.

option_prompt(*config*, *p_opt*)

Prompt user for option that requires a value

magic_fixes(*config*)

Perform the set of magic argument fixups

get_one(*cloud=None*, *validate=True*, *argparse=None*, ***kwargs*)

Retrieve a single CloudRegion and merge additional options

Parameters

- **cloud** (*string*) The name of the configuration to load from clouds.yaml
- **validate** (*boolean*) Validate the config. Setting this to False causes no auth plugin to be created. Its really only useful for testing.
- **argparse** (*Namespace*) An argparse Namespace object; allows direct passing in of argparse options to be added to the cloud config. Values of None and will be removed.
- **region_name** Name of the region of the cloud.
- **kwargs** Additional configuration options

Returns

openstack.config.cloud_region.CloudRegion

Raises

keystoneauth1.exceptions.MissingRequiredOptions on missing required auth parameters

`get_one_cloud_osc`(*cloud=None*, *validate=True*, *argparse=None*, ***kwargs*)

Retrieve a single CloudRegion and merge additional options

Parameters

- **cloud** (*string*) The name of the configuration to load from clouds.yaml
- **validate** (*boolean*) Validate the config. Setting this to False causes no auth plugin to be created. Its really only useful for testing.
- **argparse** (*Namespace*) An argparse Namespace object; allows direct passing in of argparse options to be added to the cloud config. Values of None and will be removed.
- **region_name** Name of the region of the cloud.
- **kwargs** Additional configuration options

Raises

keystoneauth1.exceptions.MissingRequiredOptions on missing required auth parameters

`static set_one_cloud`(*config_file*, *cloud*, *set_config=None*)

Set a single cloud configuration.

Parameters

- **config_file** (*string*) The path to the config file to edit. If this file does not exist it will be created.
- **cloud** (*string*) The name of the configuration to save to clouds.yaml
- **set_config** (*dict*) Configuration options to be set

`class openstack.config.cloud_region.CloudRegion`(*name=None*, *region_name=None*, *config=None*, *force_ipv4=False*, *auth_plugin=None*, *openstack_config=None*, *session_constructor=None*, *app_name=None*, *app_version=None*, *session=None*, *discovery_cache=None*, *extra_config=None*, *cache_expiration_time=0*, *cache_expirations=None*, *cache_path=None*, *cache_class='dogpile.cache.null'*, *cache_arguments=None*, *password_callback=None*, *statsd_host=None*, *statsd_port=None*, *statsd_prefix=None*, *influxdb_config=None*, *collector_registry=None*, *cache_auth=False*)

The configuration for a Region of an OpenStack Cloud.

A CloudRegion encapsulates the config information needed for connections to all of the services in a Region of a Cloud.

Parameters

- **region_name** (*str*) The default region name for all services in this CloudRegion. If both `region_name` and `config['region_name']` are specified, the kwarg takes precedence. May be overridden for a given ${service} via a ${service}_region_name key in the `config` dict.
- **config** (*dict*) A dict of configuration values for the CloudRegion and its services. The key for a ${config_option} for a specific ${service} should be ${service}_${config_option}. For example, to configure the endpoint_override for the block_storage service, the `config` dict should contain:

```
'block_storage_endpoint_override': 'http://...'
```

To provide a default to be used if no service-specific override is present, just use the unprefixed ${config_option} as the service key, e.g.:

```
'interface': 'public'
```

property full_name

Return a string that can be used as an identifier.

Always returns a valid string. It will have name and region_name or just one of the two if only one is set, or else unknown.

set_session_constructor(*session_constructor*)

Sets the Session constructor.

get_requests_verify_args()

Return the verify and cert values for the requests library.

get_services()

Return a list of service types we know something about.

get_endpoint_from_catalog(*service_type*, *interface=None*, *region_name=None*)

Return the endpoint for a given service as found in the catalog.

For values respecting endpoint overrides, see *endpoint_for()*

Parameters

- **service_type** Service Type of the endpoint to search for.
- **interface** Interface of the endpoint to search for. Optional, defaults to the configured value for interface for this Connection.
- **region_name** Region Name of the endpoint to search for. Optional, defaults to the configured value for region_name for this Connection.

Returns

The endpoint of the service, or None if not found.

get_auth()

Return a keystoneauth plugin from the auth credentials.

insert_user_agent()

Set sdk information into the user agent of the Session.

> **Warning**
>
> This method is here to be used by os-client-config. It exists as a hook point so that os-client-config can provice backwards compatibility and still be in the User Agent for people using os-client-config directly.

Normal consumers of SDK should use app_name and app_version. However, if someone else writes a subclass of *CloudRegion* it may be desirable.

get_session()

Return a keystoneauth session based on the auth credentials.

get_service_catalog()

Helper method to grab the service catalog.

get_session_client(*service_type*, *version=None*, *constructor=<class 'openstack.proxy.Proxy'>*, ***kwargs*)

Return a prepped keystoneauth Adapter for a given service.

This is useful for making direct requests calls against a mounted endpoint. That is, if you do:

client = get_session_client(compute)

then you can do:

client.get(/flavors)

and it will work like you think.

get_session_endpoint(*service_type*, *min_version=None*, *max_version=None*)

Return the endpoint from config or the catalog.

If a configuration lists an explicit endpoint for a service, return that. Otherwise, fetch the service catalog from the keystone session and return the appropriate endpoint.

Parameters

service_type Official service type of service

get_cache_resource_expiration(*resource*, *default=None*)

Get expiration time for a resource

Parameters

- **resource** Name of the resource type
- **default** Default value to return if not found (optional, defaults to None)

Returns

Expiration time for the resource type as float or default

requires_floating_ip()

Return whether or not this cloud requires floating ips.

Returns

True of False if know, None if discovery is needed. If requires_floating_ip is

not configured but the cloud is known to not provide floating ips, will return False.

get_external_networks()

Get list of network names for external networks.

get_external_ipv4_networks()

Get list of network names for external IPv4 networks.

get_external_ipv6_networks()

Get list of network names for external IPv6 networks.

get_internal_networks()

Get list of network names for internal networks.

get_internal_ipv4_networks()

Get list of network names for internal IPv4 networks.

get_internal_ipv6_networks()

Get list of network names for internal IPv6 networks.

get_default_network()

Get network used for default interactions.

get_nat_destination()

Get network used for NAT destination.

get_nat_source()

Get network used for NAT source.

get_client_config(*name=None*, *defaults=None*)

Get config settings for a named client.

Settings will also be looked for in a section called client. If settings are found in both, they will be merged with the settings from the named section winning over the settings from client section, and both winning over provided defaults.

Parameters

- **name** (*string*) Name of the config section to look for.
- **defaults** (*dict*) Default settings to use.

Returns

A dict containing merged settings from the named section, the client section and the defaults.

Connect

In order to work with an OpenStack cloud you first need to create a *Connection* to it using your credentials. A *Connection* can be created in 3 ways, using the class itself, *Config Files*, or *Environment Variables*. It is recommended to always use *Config Files* as the same config can be used across tools and languages.

Create Connection

To create a *Connection* instance, use the `connect()` factory function.

```
def create_connection(
    auth_url,
    region,
    project_name,
    username,
    password,
    user_domain,
    project_domain,
):
    return openstack.connect(
        auth_url=auth_url,
        project_name=project_name,
        username=username,
        password=password,
        region_name=region,
        user_domain_name=user_domain,
        project_domain_name=project_domain,
        app_name='examples',
        app_version='1.0',
    )
```

Full example at connect.py

Note

To enable logging, see the *Logging* user guide.

Next

Now that you can create a connection, continue with the *User Guides* to work with an OpenStack service.

Connect From Config

In order to work with an OpenStack cloud you first need to create a *Connection* to it using your credentials. A *Connection* can be created in 3 ways, using the class itself (see *Connect*), a file, or environment variables as illustrated below. The SDK uses os-client-config to handle the configuration.

Create Connection From A File

Default Location

To create a connection from a file you need a YAML file to contain the configuration.

```
clouds:
  devstack:
    auth:
      auth_url: http://xxx.xxx.xxx.xxx/identity
```

(continues on next page)

(continued from previous page)

```
      password: password
      project_domain_id: default
      project_name: demo
      user_domain_id: default
      username: demo
    identity_api_version: '3'
    region_name: RegionOne
    volume_api_version: '3'
  devstack-admin:
    auth:
      auth_url: http://xxx.xxx.xxx.xxx/identity
      password: password
      project_domain_id: default
      project_name: admin
      user_domain_id: default
      username: admin
    identity_api_version: '3'
    region_name: RegionOne
    volume_api_version: '3'
  devstack-alt:
    auth:
      auth_url: http://xxx.xxx.xxx.xxx/identity
      password: password
      project_domain_id: default
      project_name: alt_demo
      user_domain_id: default
      username: alt_demo
    identity_api_version: '3'
    region_name: RegionOne
    volume_api_version: '3'
example:
  image_name: cirros-0.5.2-x86_64-disk
  flavor_name: m1.small
```

To use a configuration file called `clouds.yaml` in one of the default locations:

- Current Directory
- ~/.config/openstack
- /etc/openstack

call *from_config()*. The `from_config` function takes three optional arguments:

- **cloud_name** allows you to specify a cloud from your `clouds.yaml` file.
- **cloud_config** allows you to pass in an existing `openstack.config.loader.OpenStackConfig`` object.
- **options** allows you to specify a namespace object with options to be added to the cloud config.

```
class Opts:
    def __init__(self, cloud_name='devstack-admin', debug=False):
```

(continues on next page)

(continued from previous page)

```
        self.cloud = cloud_name
        self.debug = debug
        # Use identity v3 API for examples.
        self.identity_api_version = '3'
```

```
def create_connection_from_config():
    return openstack.connect(cloud=TEST_CLOUD)
```

```
def create_connection_from_args():
    parser = argparse.ArgumentParser()
    return openstack.connect(options=parser)
```

Note

To enable logging, set `debug=True` in the `options` object.

User Defined Location

To use a configuration file in a user defined location set the environment variable `OS_CLIENT_CONFIG_FILE` to the absolute path of a file.:

```
export OS_CLIENT_CONFIG_FILE=/path/to/my/config/my-clouds.yaml
```

and call *from_config()* with the **cloud_name** of the cloud configuration to use, .

Next

Now that you can create a connection, continue with the *User Guides* for an OpenStack service.

Logging

Note

TODO(shade) This document is written from a shade POV. It needs to be combined with the existing logging guide, but also the logging systems need to be rationalized.

openstacksdk uses *Python Logging*. As *openstacksdk* is a library, it does not configure logging handlers automatically, expecting instead for that to be the purview of the consuming application.

Simple Usage

For consumers who just want to get a basic logging setup without thinking about it too deeply, there is a helper method. If used, it should be called before any other openstacksdk functionality.

`openstack.enable_logging`(*debug=False*, *http_debug=False*, *path=None*, *stream=None*, *format_stream=False*, *format_template='%(asctime)s %(levelname)s: %(name)s %(message)s'*, *handlers=None*)

Enable logging output.

Helper function to enable logging. This function is available for debugging purposes and for folks doing simple applications who want an easy just make it work for me. For more complex applications or for those who want more flexibility, the standard library `logging` package will receive these messages in any handlers you create.

Parameters

- **debug** (*bool*) Set this to `True` to receive debug messages.
- **http_debug** (*bool*) Set this to `True` to receive debug messages including HTTP requests and responses. This implies `debug=True`.
- **path** (*str*) If a *path* is specified, logging output will written to that file in addition to sys.stderr. The path is passed to logging.FileHandler, which will append messages the file (and create it if needed).
- **stream** One of `None `` or ``sys.stdout` or `sys.stderr`. If it is `None`, nothing is logged to a stream. If it isnt `None`, console output is logged to this stream.
- **format_stream** (*bool*) If format_stream is False, the default, apply `format_template` to `path` but not to `stream` outputs. If True, apply `format_template` to `stream` outputs as well.
- **format_template** (*str*) Template to pass to `logging.Formatter`.

Return type
None

```
import openstack
openstack.enable_logging()
```

The `stream` parameter controls the stream where log message are written to. It defaults to *sys.stdout* which will result in log messages being written to STDOUT. It can be set to another output stream, or to `None` to disable logging to the console.

The `path` parameter sets up logging to log to a file. By default, if `path` is given and `stream` is not, logging will only go to `path`.

You can combine the `path` and `stream` parameters to log to both places simultaneously.

To log messages to a file called `openstack.log` and the console on `stdout`:

```
import sys
import openstack

openstack.enable_logging(
    debug=True, path='openstack.log', stream=sys.stdout)
```

openstack.enable_logging also sets up a few other loggers and squelches some warnings or log messages that are otherwise uninteresting or unactionable by an openstacksdk user.

Advanced Usage

openstacksdk logs to a set of different named loggers.

Most of the logging is set up to log to the root `openstack` logger. There are additional sub-loggers that are used at times, primarily so that a user can decide to turn on or off a specific type of logging. They are listed below.

openstack.config
: Issues pertaining to configuration are logged to the `openstack.config` logger.

openstack.iterate_timeout
: When *openstacksdk* needs to poll a resource, it does so in a loop that waits between iterations and ultimately times out. The `openstack.iterate_timeout` logger emits messages for each iteration indicating it is waiting and for how long. These can be useful to see for long running tasks so that one can know things are not stuck, but can also be noisy.

openstack.fnmatch
: *openstacksdk* will try to use fnmatch on given *name_or_id* arguments. Its a best effort attempt, so pattern misses are logged to `openstack.fnmatch`. A user may not be intending to use an fnmatch pattern - such as if they are trying to find an image named `Fedora 24 [official]`, so these messages are logged separately.

HTTP Tracing

HTTP Interactions are handled by keystoneauth. If you want to enable HTTP tracing while using openstacksdk and are not using *openstack.enable_logging*, set the log level of the `keystoneauth` logger to `DEBUG`.

For more information see https://docs.openstack.org/keystoneauth/latest/using-sessions.html#logging

Python Logging

Python logging is a standard feature of Python and is documented fully in the Python Documentation, which varies by version of Python.

For more information on Python Logging for Python v2, see https://docs.python.org/2/library/logging.html.

For more information on Python Logging for Python v3, see https://docs.python.org/3/library/logging.html.

Statistics reporting

openstacksdk can report statistics on individual API requests/responses in several different formats.

Note that metrics will be reported only when corresponding client libraries (*statsd* for statsd reporting, *influxdb* for influxdb, etc.). If libraries are not available reporting will be silently ignored.

statsd

statsd can be configured via configuration entries or environment variables.

A global *metrics* entry defines defaults for all clouds. Each cloud can specify a *metrics* section to override variables; this may be useful to separate results reported for each cloud.

```
metrics:
  statsd:
    host: __statsd_server_host__
```

(continues on next page)

(continued from previous page)

```
    port: __statsd_server_port__
    prefix: __statsd_prefix__ (default 'openstack.api')
clouds:
  a-cloud:
    auth:
      ...
    metrics:
      statsd:
        prefix: 'openstack.api.a-cloud'
```

If the *STATSD_HOST* or *STATSD_PORT* environment variables are set, they will be taken as the default values (and enable *statsd* reporting if no other configuration is specified).

InfluxDB

InfluxDB is supported via configuration in the *metrics* field. Similar to *statsd*, each cloud can provide its own *metrics* section to override any global defaults.

```
metrics:
  influxdb:
    host: __influxdb_server_host__
    port: __influxdb_server_port__
    use_udp: __True|False__
    username: __influxdb_auth_username__
    password: __influxdb_auth_password__
    database: __influxdb_db_name__
    measurement: __influxdb_measurement_name__
    timeout: __infludb_requests_timeout__
clouds:
  ..
```

InfluxDB reporting allows setting additional tags into the metrics based on the selected cloud.

```
clouds:
  my_cloud:
    profile: some_profile
    ...
    additional_metric_tags:
      environment: production
```

prometheus

The prometheus support does not read from config, and does not run an http service since OpenstackSDK is a library. It is expected that an application that uses OpenstackSDK and wants request stats be collected will pass a *prometheus_client.CollectorRegistry* to *collector_registry*.

Microversions

As openstacksdk rolls out support for consuming microversions, it will do so on a call by call basis as needed. Just like with major versions, openstacksdk should have logic to handle each microversion for a given REST call it makes, with the following rules in mind:

- If an activity openstack performs can be done differently or more efficiently with a new microversion, the support should be added to openstack.cloud and to the appropriate Proxy class.
- openstacksdk should always attempt to use the latest microversion it is aware of for a given call, unless a microversion removes important data.
- Microversion selection should under no circumstances be exposed to the user in python API calls in the Resource layer or the openstack.cloud layer.
- Microversion selection is exposed to the user in the REST layer via the `microversion` argument to each REST call.
- A user of the REST layer may set the default microversion by setting `{service_type}_default_microversion` in clouds.yaml or `OS_{service_type|upper}_DEFAULT_MICROVERSION` environment variable.

Note

Setting the default microversion in any circumstance other than when using the REST layer is highly discouraged. Both of the higher layers in openstacksdk provide data normalization as well as logic about which REST call to make. Setting the default microversion could change the behavior of the service in question in such a way that openstacksdk does not understand. If there is a feature of a service that needs a microversion and it is not already transparently exposed in openstacksdk, please file a bug.

- If a feature is only exposed for a given microversion and cannot be simulated for older clouds without that microversion, it is ok to add it, but a clear error message should be given to the user that the given feature is not available on their cloud. (A message such as This cloud supports a maximum microversion of XXX for service YYY and this feature only exists on clouds with microversion ZZZ. Please contact your cloud provider for information about when this feature might be available)
- When adding a feature that only exists behind a new microversion, every effort should be made to figure out how to provide the same functionality if at all possible, even if doing so is inefficient. If an inefficient workaround is employed, a warning should be provided to the user. (the users workaround to skip the inefficient behavior would be to stop using that openstacksdk API call) An example of this is the nova get me a network feature. The logic of get me a network can be done client-side, albeit less efficiently. Adding support for the get me a network feature via nova microversion should also add support for doing the client-side workaround.
- If openstacksdk is aware of logic for more than one microversion, it should always attempt to use the latest version available for the service for that call.
- Objects returned from openstacksdk should always go through normalization and thus should always conform to openstacksdks documented data model. The objects should never look different to the user regardless of the microversion used for the REST call.
- If a microversion adds new fields to an object, those fields should be added to openstacksdks data model contract for that object and the data should either be filled in by performing additional REST

calls if the data is available that way, or the field should have a default value of None which the user can be expected to test for when attempting to use the new value.

- If a microversion removes fields from an object that are part of the existing data model contract, care should be taken to not use the new microversion for that call unless forced to by lack of availablity of the old microversion on the cloud in question. In the case where an old microversion is no longer available, care must be taken to either find the data from another source and fill it in, or to put a value of None into the field and document for the user that on some clouds the value may not exist.
- If a microversion removes a field and the outcome is particularly intractable and impossible to work around without fundamentally breaking users, an issue should be raised with the service team in question. Hopefully a resolution can be found during the period while clouds still have the old microversion.
- As new calls or objects are added, it is important to check in with the service team in question on the expected stability of the object. If there are known changes expected in the future, even if they may be a few years off, openstacksdk should take care to not add committments to its data model for those fields/features. It is ok for openstacksdk to not have something.

Note

openstacksdk does not currently have any sort of experimental opt-in API that would allow exposing things to a user that may not be supportable under the normal compatibility contract. If a conflict arises in the future where there is a strong desire for a feature but also a lack of certainty about its stability over time, an experimental API may want to be explored but concrete use cases should arise before such a thing is started.

Using OpenStack Baremetal

Before working with the Bare Metal service, youll need to create a connection to your OpenStack cloud by following the *Connect* user guide. This will provide you with the `conn` variable used in the examples below.

Table of Contents

The primary resource of the Bare Metal service is the **node**.

Below are a few usage examples. For a reference to all the available methods, see *Baremetal API*.

CRUD operations

List Nodes

A **node** is a bare metal machine.

```
def list_nodes(conn):
    print("List Nodes:")

    for node in conn.baremetal.nodes():
        print(node)
```

Full example: baremetal resource list

Provisioning operations

Provisioning actions are the main way to manipulate the nodes. See Bare Metal service states documentation for details.

Manage and inspect Node

Managing a node in the `enroll` provision state validates the management (IPMI, Redfish, etc) credentials and moves the node to the `manageable` state. *Managing* a node in the `available` state moves it to the `manageable` state. In this state additional actions, such as configuring RAID or inspecting, are available.

Inspecting a node detects its properties by either talking to its BMC or by booting a special ramdisk.

```
def manage_and_inspect_node(conn, uuid):
    node = conn.baremetal.find_node(uuid)
    print('Before:', node.provision_state)
    conn.baremetal.set_node_provision_state(node, 'manage')
    conn.baremetal.wait_for_nodes_provision_state([node], 'manageable')
    conn.baremetal.set_node_provision_state(node, 'inspect')
    res = conn.baremetal.wait_for_nodes_provision_state([node], 'manageable')
    print('After:', res[0].provision_state)
```

Full example: baremetal provisioning

Provide Node

Providing a node in the `manageable` provision state makes it available for deployment.

```
def provide_node(conn, uuid):
    node = conn.baremetal.find_node(uuid)
    print('Before:', node.provision_state)
    conn.baremetal.set_node_provision_state(node, 'provide')
    res = conn.baremetal.wait_for_nodes_provision_state([node], 'available')
    print('After:', res[0].provision_state)
```

Full example: baremetal provisioning

Using OpenStack Block Storage

Before working with the Block Storage service, youll need to create a connection to your OpenStack cloud by following the *Connect* user guide. This will provide you with the `conn` variable used in the examples below.

Using OpenStack Clustering

Before working with the Clustering service, youll need to create a connection to your OpenStack cloud by following the *Connect* user guide. This will provide you with the `conn` variable used by all examples in this guide.

The primary abstractions/resources of the Clustering service are:

Working with Profile Types

A **profile** is a template used to create and manage nodes, i.e. objects exposed by other OpenStack services. A profile encodes the information needed for node creation in a property named `spec`.

List Profile Types

To examine the known profile types:

```
def list_profile_types(conn):
    print("List Profile Types:")

    for pt in conn.clustering.profile_types():
        print(pt.to_dict())
```

Full example: manage profile type

Get Profile Type

To get the details about a profile type, you need to provide the name of it.

```
def get_profile_type(conn):
    print("Get Profile Type:")

    pt = conn.clustering.get_profile_type('os.nova.server-1.0')

    print(pt.to_dict())
```

Full example: manage profile type

Managing Profiles

A **profile type** can be treated as the meta-type of a *Profile* object. A registry of profile types is built when the Cluster service starts. When creating a *Profile* object, you will indicate the profile type used in its *spec* property.

List Profiles

To examine the list of profiles:

```
def list_profiles(conn):
    print("List Profiles:")

    for profile in conn.clustering.profiles():
        print(profile.to_dict())

    for profile in conn.clustering.profiles(sort='name:asc'):
        print(profile.to_dict())
```

When listing profiles, you can specify the sorting option using the `sort` parameter and you can do pagination using the `limit` and `marker` parameters.

Full example: manage profile

Create Profile

When creating a profile, you will provide a dictionary with keys and values specified according to the profile type referenced.

```
def create_profile(conn):
    print("Create Profile:")

    spec = {
        'profile': 'os.nova.server',
        'version': 1.0,
        'name': 'os_server',
        'properties': {
            'name': SERVER_NAME,
            'flavor': FLAVOR_NAME,
            'image': IMAGE_NAME,
            'networks': {'network': NETWORK_NAME},
        },
    }

    profile = conn.clustering.create_profile(spec)
    print(profile.to_dict())
```

Optionally, you can specify a `metadata` keyword argument that contains some key-value pairs to be associated with the profile.

Full example: manage profile

Find Profile

To find a profile based on its name or ID:

```
def find_profile(conn):
    print("Find Profile:")
```

(continues on next page)

(continued from previous page)

```
    profile = conn.clustering.find_profile('os_server')
    print(profile.to_dict())
```

The Cluster service doesnt allow updating the `spec` of a profile. The only way to achieve that is to create a new profile.

Full example: manage profile

Get Profile

To get a profile based on its name or ID:

```
def get_profile(conn):
    print("Get Profile:")

    profile = conn.clustering.get_profile('os_server')
    print(profile.to_dict())
```

Full example: manage profile

Update Profile

After a profile is created, most of its properties are immutable. Still, you can update a profiles `name` and/or `metadata`.

```
def update_profile(conn):
    print("Update Profile:")

    profile = conn.clustering.update_profile('os_server', name='old_server')
    print(profile.to_dict())
```

The Cluster service doesnt allow updating the `spec` of a profile. The only way to achieve that is to create a new profile.

Full example: manage profile

Delete Profile

A profile can be deleted after creation, provided that it is not referenced by any active clusters or nodes. If you attempt to delete a profile that is still in use, you will get an error message.

```
def delete_profile(conn):
    print("Delete Profile:")

    conn.clustering.delete_profile('os_server')

    print("Profile deleted.")
```

Managing Clusters

Clusters are first-class citizens in Senlin service design. A cluster is defined as a collection of homogeneous objects. The homogeneous here means that the objects managed (aka. Nodes) have to be instantiated from the same profile type.

List Clusters

To examine the list of receivers:

```
def list_cluster(conn):
    print("List clusters:")

    for cluster in conn.clustering.clusters():
        print(cluster.to_dict())

    for cluster in conn.clustering.clusters(sort='name:asc'):
        print(cluster.to_dict())
```

When listing clusters, you can specify the sorting option using the `sort` parameter and you can do pagination using the `limit` and `marker` parameters.

Full example: manage cluster

Create Cluster

When creating a cluster, you will provide a dictionary with keys and values according to the cluster type referenced.

```
def create_cluster(conn):
    print("Create cluster:")

    spec = {
        "name": CLUSTER_NAME,
        "profile_id": PROFILE_ID,
        "min_size": 0,
        "max_size": -1,
        "desired_capacity": 1,
    }

    cluster = conn.clustering.create_cluster(**spec)
    print(cluster.to_dict())
```

Optionally, you can specify a `metadata` keyword argument that contains some key-value pairs to be associated with the cluster.

Full example: manage cluster

Get Cluster

To get a cluster based on its name or ID:

```
def get_cluster(conn):
    print("Get cluster:")

    cluster = conn.clustering.get_cluster(CLUSTER_ID)
    print(cluster.to_dict())
```

Full example: manage cluster

Find Cluster

To find a cluster based on its name or ID:

```
def find_cluster(conn):
    print("Find cluster:")

    cluster = conn.clustering.find_cluster(CLUSTER_ID)
    print(cluster.to_dict())
```

Full example: manage cluster

Update Cluster

After a cluster is created, most of its properties are immutable. Still, you can update a clusters `name` and/or `params`.

```
def update_cluster(conn):
    print("Update cluster:")

    spec = {
        "name": "Test_Cluster001",
        "profile_id": "c0e3a680-e270-4eb8-9361-e5c9503fba0a",
        "profile_only": True,
    }
    cluster = conn.clustering.update_cluster(CLUSTER_ID, **spec)
    print(cluster.to_dict())
```

Full example: manage cluster

Delete Cluster

A cluster can be deleted after creation, When there are nodes in the cluster, the Senlin engine will launch a process to delete all nodes from the cluster and destroy them before deleting the cluster object itself.

```
def delete_cluster(conn):
    print("Delete cluster:")

    conn.clustering.delete_cluster(CLUSTER_ID)
    print("Cluster deleted.")

    # cluster support force delete
```

(continues on next page)

(continued from previous page)

```
    conn.clustering.delete_cluster(CLUSTER_ID, False, True)
    print("Cluster deleted")
```

Add Nodes to Cluster

Add some existing nodes into the specified cluster.

```
def add_nodes_to_cluster(conn):
    print("Add nodes to cluster:")

    node_ids = [NODE_ID]
    res = conn.clustering.add_nodes_to_cluster(CLUSTER_ID, node_ids)
    print(res)
```

Remove Nodes from Cluster

Remove nodes from specified cluster.

```
def remove_nodes_from_cluster(conn):
    print("Remove nodes from a cluster:")

    node_ids = [NODE_ID]
    res = conn.clustering.remove_nodes_from_cluster(CLUSTER_ID, node_ids)
    print(res)
```

Replace Nodes in Cluster

Replace some existing nodes in the specified cluster.

```
def replace_nodes_in_cluster(conn):
    print("Replace the nodes in a cluster with specified nodes:")

    old_node = NODE_ID
    new_node = "cd803d4a-015d-4223-b15f-db29bad3146c"
    spec = {old_node: new_node}
    res = conn.clustering.replace_nodes_in_cluster(CLUSTER_ID, **spec)
    print(res)
```

Cluster Scale Out

Inflate the size of a cluster.

```
def scale_out_cluster(conn):
    print("Inflate the size of a cluster:")

    res = conn.clustering.scale_out_cluster(CLUSTER_ID, 1)
    print(res)
```

Cluster Scale In

Shrink the size of a cluster.

```
def scale_out_cluster(conn):
    print("Inflate the size of a cluster:")

    res = conn.clustering.scale_out_cluster(CLUSTER_ID, 1)
    print(res)
```

Cluster Resize

Resize of cluster.

```
def resize_cluster(conn):
    print("Resize of cluster:")

    spec = {
        'min_size': 1,
        'max_size': 6,
        'adjustment_type': 'EXACT_CAPACITY',
        'number': 2,
    }
    res = conn.clustering.resize_cluster(CLUSTER_ID, **spec)
    print(res)
```

Attach Policy to Cluster

Once a policy is attached (bound) to a cluster, it will be enforced when related actions are performed on that cluster, unless the policy is (temporarily) disabled on the cluster

```
def attach_policy_to_cluster(conn):
    print("Attach policy to a cluster:")

    spec = {'enabled': True}
    res = conn.clustering.attach_policy_to_cluster(
        CLUSTER_ID, POLICY_ID, **spec
    )
    print(res)
```

Detach Policy from Cluster

Once a policy is attached to a cluster, it can be detached from the cluster at users request.

```
def detach_policy_from_cluster(conn):
    print("Detach a policy from a cluster:")

    res = conn.clustering.detach_policy_from_cluster(CLUSTER_ID, POLICY_ID)
    print(res)
```

Cluster Check

Check cluster health status, Cluster members can be check.

```
def check_cluster(conn):
    print("Check cluster:")

    res = conn.clustering.check_cluster(CLUSTER_ID)
    print(res)
```

Cluster Recover

To restore a specified cluster, members in the cluster will be checked.

```
def recover_cluster(conn):
    print("Recover cluster:")

    spec = {'check': True}
    res = conn.clustering.recover_cluster(CLUSTER_ID, **spec)
    print(res)
```

Managing Nodes

Node is a logical object managed by the Senlin service. A node can be a member of at most one cluster at any time. A node can be an orphan node which means it doesnt belong to any clusters.

List Nodes

To examine the list of Nodes:

```
def list_nodes(conn):
    print("List Nodes:")

    for node in conn.clustering.nodes():
        print(node.to_dict())
    for node in conn.clustering.nodes(sort='asc:name'):
        print(node.to_dict())
```

When listing nodes, you can specify the sorting option using the `sort` parameter and you can do pagination using the `limit` and `marker` parameters.

Full example: manage node

Create Node

When creating a node, you will provide a dictionary with keys and values according to the node type referenced.

```
def create_node(conn):
    print("Create Node:")

    spec = {
```

(continues on next page)

(continued from previous page)

```
        'name': NODE_NAME,
        'profile_id': PROFILE_ID,
    }
    node = conn.clustering.create_node(**spec)
    print(node.to_dict())
```

Optionally, you can specify a `metadata` keyword argument that contains some key-value pairs to be associated with the node.

Full example: manage node

Get Node

To get a node based on its name or ID:

```
def get_node(conn):
    print("Get Node:")

    node = conn.clustering.get_node(NODE_ID)
    print(node.to_dict())
```

Full example: manage node

Find Node

To find a node based on its name or ID:

```
def find_node(conn):
    print("Find Node:")

    node = conn.clustering.find_node(NODE_ID)
    print(node.to_dict())
```

Full example: manage node

Update Node

After a node is created, most of its properties are immutable. Still, you can update a nodes `name` and/or `params`.

```
def update_node(conn):
    print("Update Node:")

    spec = {
        'name': 'Test_Node01',
        'profile_id': 'c0e3a680-e270-4eb8-9361-e5c9503fba0b',
    }

    node = conn.clustering.update_node(NODE_ID, **spec)
    print(node.to_dict())
```

Full example: manage node

Delete Node

A node can be deleted after creation, provided that it is not referenced by any active clusters. If you attempt to delete a node that is still in use, you will get an error message.

```
def delete_node(conn):
    print("Delete Node:")

    conn.clustering.delete_node(NODE_ID)
    print("Node deleted.")
    # node support force delete
    conn.clustering.delete_node(NODE_ID, False, True)
    print("Node deleted")
```

Full example: manage node

Check Node

If the underlying physical resource is not healthy, the node will be set to ERROR status.

```
def check_node(conn):
    print("Check Node:")

    node = conn.clustering.check_node(NODE_ID)
    print(node)
```

Full example: manage node

Recover Node

To restore a specified node.

```
def recover_node(conn):
    print("Recover Node:")

    spec = {'check': True}
    node = conn.clustering.recover_node(NODE_ID, **spec)
    print(node)
```

Working with Policy Types

A **policy** is a template that encodes the information needed for specifying the rules that are checked/enforced before/after certain actions are performed on a cluster. The rules are encoded in a property named `spec`.

List Policy Types

To examine the known policy types:

```
def list_policy_types(conn):
    print("List Policy Types:")

```

(continues on next page)

(continued from previous page)

```
    for pt in conn.clustering.policy_types():
        print(pt.to_dict())
```

Full example: manage policy type

Get Policy Type

To retrieve the details about a policy type, you need to provide the name of it.

```
def get_policy_type(conn):
    print("Get Policy Type:")

    pt = conn.clustering.get_policy_type('senlin.policy.deletion-1.0')

    print(pt.to_dict())
```

Full example: manage policy type

Managing Policies

A **policy type** can be treated as the meta-type of a *Policy* object. A registry of policy types is built when the Cluster service starts. When creating a *Policy* object, you will indicate the policy type used in its *spec* property.

List Policies

To examine the list of policies:

```
def list_policies(conn):
    print("List Policies:")

    for policy in conn.clustering.policies():
        print(policy.to_dict())

    for policy in conn.clustering.policies(sort='name:asc'):
        print(policy.to_dict())
```

When listing policies, you can specify the sorting option using the `sort` parameter and you can do pagination using the `limit` and `marker` parameters.

Full example: manage policy

Create Policy

When creating a policy, you will provide a dictionary with keys and values according to the policy type referenced.

```
def create_policy(conn):
    print("Create Policy:")
    attrs = {
        'name': 'dp01',
```

(continues on next page)

(continued from previous page)

```
        'spec': {
            'policy': 'senlin.policy.deletion',
            'version': 1.0,
            'properties': {
                'criteria': 'oldest_first',
                'destroy_after_deletion': True,
            },
        },
    }

    policy = conn.clustering.create_policy(attrs)
    print(policy.to_dict())
```

Optionally, you can specify a `metadata` keyword argument that contains some key-value pairs to be associated with the policy.

Full example: manage policy

Find Policy

To find a policy based on its name or ID:

```
def find_policy(conn):
    print("Find Policy:")

    policy = conn.clustering.find_policy('dp01')
    print(policy.to_dict())
```

Full example: manage policy

Get Policy

To get a policy based on its name or ID:

```
def get_policy(conn):
    print("Get Policy:")

    policy = conn.clustering.get_policy('dp01')
    print(policy.to_dict())
```

Full example: manage policy

Update Policy

After a policy is created, most of its properties are immutable. Still, you can update a policys `name` and/or `metadata`.

```
def update_policy(conn):
    print("Update Policy:")

```

(continues on next page)

(continued from previous page)

```
    policy = conn.clustering.update_policy('dp01', name='dp02')
    print(policy.to_dict())
```

The Cluster service doesnt allow updating the `spec` of a policy. The only way to achieve that is to create a new policy.

Full example: manage policy

Delete Policy

A policy can be deleted after creation, provided that it is not referenced by any active clusters or nodes. If you attempt to delete a policy that is still in use, you will get an error message.

```
def delete_policy(conn):
    print("Delete Policy:")

    conn.clustering.delete_policy('dp01')

    print("Policy deleted.")
```

Managing Receivers

Receivers are the event sinks associated to senlin clusters. When certain events (or alarms) are seen by a monitoring software, the software can notify the senlin clusters of those events (or alarms). When senlin receives those notifications, it can automatically trigger some predefined operations with preset parameter values.

List Receivers

To examine the list of receivers:

```
def list_receivers(conn):
    print("List Receivers:")

    for receiver in conn.clustering.receivers():
        print(receiver.to_dict())

    for receiver in conn.clustering.receivers(sort='name:asc'):
        print(receiver.to_dict())
```

When listing receivers, you can specify the sorting option using the `sort` parameter and you can do pagination using the `limit` and `marker` parameters.

Full example: manage receiver

Create Receiver

When creating a receiver, you will provide a dictionary with keys and values according to the receiver type referenced.

```
def create_receiver(conn):
    print("Create Receiver:")

    # Build the receiver attributes and create the recever.
    spec = {
        "action": "CLUSTER_SCALE_OUT",
        "cluster_id": CLUSTER_ID,
        "name": FAKE_NAME,
        "params": {"count": "1"},
        "type": "webhook",
    }

    receiver = conn.clustering.create_receiver(**spec)
    print(receiver.to_dict())
```

Optionally, you can specify a `metadata` keyword argument that contains some key-value pairs to be associated with the receiver.

Full example: manage receiver

Get Receiver

To get a receiver based on its name or ID:

```
def get_receiver(conn):
    print("Get Receiver:")

    receiver = conn.clustering.get_receiver(FAKE_NAME)
    print(receiver.to_dict())
```

Full example: manage receiver

Find Receiver

To find a receiver based on its name or ID:

```
def find_receiver(conn):
    print("Find Receiver:")

    receiver = conn.clustering.find_receiver(FAKE_NAME)
    print(receiver.to_dict())
```

Full example: manage receiver

Update Receiver

After a receiver is created, most of its properties are immutable. Still, you can update a receivers `name` and/or `params`.

```
def update_receiver(conn):
    print("Update Receiver:")
```

(continues on next page)

(continued from previous page)

```
    spec = {"name": "test_receiver2", "params": {"count": "2"}}
    receiver = conn.clustering.update_receiver(FAKE_NAME, **spec)
    print(receiver.to_dict())
```

Full example: manage receiver

Delete Receiver

A receiver can be deleted after creation, provided that it is not referenced by any active clusters. If you attempt to delete a receiver that is still in use, you will get an error message.

```
def delete_receiver(conn):
    print("Delete Receiver:")

    conn.clustering.delete_receiver(FAKE_NAME)
    print("Receiver deleted.")
```

Working with Actions

An action is an abstraction of some logic that can be executed by a worker thread. Most of the operations supported by Senlin are executed asynchronously, which means they are queued into database and then picked up by certain worker thread for execution.

List Actions

To examine the list of actions:

```
def list_actions(conn):
    print("List Actions:")

    for actions in conn.clustering.actions():
        print(actions.to_dict())

    for actions in conn.clustering.actions(sort='name:asc'):
        print(actions.to_dict())
```

When listing actions, you can specify the sorting option using the `sort` parameter and you can do pagination using the `limit` and `marker` parameters.

Full example: manage action

Get Action

To get a action based on its name or ID:

```
def get_action(conn):
    print("Get Action:")

    action = conn.clustering.get_action(ACTION_ID)
    print(action.to_dict())
```

Working with Events

An event is a record generated during engine execution. Such an event captures what has happened inside the senlin-engine. The senlin-engine service generates event records when it is performing some actions or checking policies.

List Events

To examine the list of events:

```
def list_events(conn):
    print("List Events:")

    for events in conn.clustering.events():
        print(events.to_dict())

    for events in conn.clustering.events(sort='name:asc'):
        print(events.to_dict())
```

When listing events, you can specify the sorting option using the `sort` parameter and you can do pagination using the `limit` and `marker` parameters.

Full example: manage event

Get Event

To get a event based on its name or ID:

```
def get_event(conn):
    print("Get Event:")

    event = conn.clustering.get_event(EVENT_ID)
    print(event.to_dict())
```

Using OpenStack Compute

Before working with the Compute service, youll need to create a connection to your OpenStack cloud by following the *Connect* user guide. This will provide you with the `conn` variable used in the examples below.

Table of Contents

The primary resource of the Compute service is the server.

List Servers

A **server** is a virtual machine that provides access to a compute instance being run by your cloud provider.

```
def list_servers(conn):
    print("List Servers:")

    for server in conn.compute.servers():
        print(server)
```

Full example: compute resource list

List Images

An **image** is the operating system you want to use for your server.

```
def list_images(conn):
    print("List Images:")

    for image in conn.compute.images():
        print(image)
```

Full example: compute resource list

List Flavors

A **flavor** is the resource configuration for a server. Each flavor is a unique combination of disk, memory, vCPUs, and network bandwidth.

```
def list_flavors(conn):
    print("List Flavors:")

    for flavor in conn.compute.flavors():
        print(flavor)
```

Full example: compute resource list

List Networks

A **network** provides connectivity to servers.

```
def list_networks(conn):
    print("List Networks:")

    for network in conn.network.networks():
        print(network)
```

Full example: network resource list

Create Key Pair

A **key pair** is the public key and private key of publickey cryptography. They are used to encrypt and decrypt login information when connecting to your server.

```
def create_keypair(conn):
    keypair = conn.compute.find_keypair(KEYPAIR_NAME)

    if not keypair:
        print("Create Key Pair:")

        keypair = conn.compute.create_keypair(name=KEYPAIR_NAME)

        print(keypair)

        try:
            os.mkdir(SSH_DIR)
        except OSError as e:
            if e.errno != errno.EEXIST:
                raise e

        with open(PRIVATE_KEYPAIR_FILE, 'w') as f:
            f.write(str(keypair.private_key))

        os.chmod(PRIVATE_KEYPAIR_FILE, 0o400)

    return keypair
```

Full example: compute resource create

Create Server

At minimum, a server requires a name, an image, a flavor, and a network on creation. You can discover the names and IDs of these attributes by listing them as above and then using the find methods to get the appropriate resources.

Ideally youll also create a server using a keypair so you can login to that server with the private key.

Servers take time to boot so we call `wait_for_server` to wait for it to become active.

```
def create_server(conn):
    print("Create Server:")

    image = conn.image.find_image(IMAGE_NAME)
    flavor = conn.compute.find_flavor(FLAVOR_NAME)
    network = conn.network.find_network(NETWORK_NAME)
    keypair = create_keypair(conn)

    server = conn.compute.create_server(
        name=SERVER_NAME,
        image_id=image.id,
        flavor_id=flavor.id,
```

(continues on next page)

(continued from previous page)

```
        networks=[{"uuid": network.id}],
        key_name=keypair.name,
    )

    server = conn.compute.wait_for_server(server)

    print(f"ssh -i {PRIVATE_KEYPAIR_FILE} root@{server.access_ipv4}")
```

Full example: compute resource create

Using OpenStack Database

Before working with the Database service, youll need to create a connection to your OpenStack cloud by following the *Connect* user guide. This will provide you with the `conn` variable used in the examples below.

Using OpenStack DNS

Before working with the DNS service, youll need to create a connection to your OpenStack cloud by following the *Connect* user guide. This will provide you with the `conn` variable used in the examples below.

Table of Contents

The primary resource of the DNS service is the server.

List Zones

Zone is a logical grouping of DNS records for a domain, allowing for the centralized management of DNS resources, including domain names, nameservers, and DNS queries.

```
def list_zones(conn):
    print("List Zones:")

    for zone in conn.dns.zones():
        print(zone)
```

Full example: dns resource list

List Recordsets

Recordsets allow for the centralized management of various DNS records within a Zone, helping to define how a domain responds to different types of DNS queries.

```
def list_recordsets(conn, name_or_id):
    print("List Recordsets for Zone")

    zone = conn.dns.find_zone(name_or_id)

    if zone:
        zone_id = zone.id
        recordsets = conn.dns.recordsets(zone_id)

        for recordset in recordsets:
            print(recordset)
    else:
        print("Zone not found.")
```

Full example: dns resource list

Create Zone

Create a zone. It allows users to define and manage the DNS namespace for a particular domain.

```
def create_zone(
    conn,
    name,
    email,
    ttl=3600,
    description="Default description",
    zone_type="PRIMARY",
):
    print("Create Zone: ")

    zone = {
        "name": name,
        "email": email,
        "ttl": ttl,
        "description": description,
        "type": zone_type,
    }

    print(conn.dns.create_zone(**zone))
```

Full example: dns resource list

Create Recordset

Create a recordset. It accepts several parameters that define the DNS records properties and sends an API request to OpenStack to create the recordset within a specified DNS zone.

```
def create_recordset(
    conn,
    name_or_id,
    recordset_name,
    recordset_type="A",
    records=["192.168.1.1"],
    ttl=3600,
    description="Default description",
):
    print("Create Recordset: ")

    zone = conn.dns.find_zone(name_or_id)

    if not zone:
        print("Zone not found.")
        return None

    zone_id = zone.id

    recordset_data = {
        "name": recordset_name,
        "type": recordset_type,
        "records": records,
        "ttl": ttl,
        "description": description,
    }

    print(conn.dns.create_recordset(zone_id, **recordset_data))
```

Full example: dns resource list

Delete Zone

Delete a zone. It allows users to completely delete the DNS management for a specified domain.

```
def delete_zone(conn, name_or_id):
    print(f"Delete Zone: {name_or_id}")

    zone = conn.dns.find_zone(name_or_id)

    if zone:
        conn.dns.delete_zone(zone.id)
    else:
        return None
```

Full example: dns resource list

Delete Recordset

Delete a recordset.

```
def delete_recordset(conn, name_or_id, recordset_name):
    print(f"Deleting Recordset: {recordset_name} in Zone: {name_or_id}")

    zone = conn.dns.find_zone(name_or_id)

    if zone:
        try:
            recordset = conn.dns.find_recordset(zone.id, recordset_name)
            if recordset:
                conn.dns.delete_recordset(recordset, zone.id)
            else:
                print("Recordset not found")
        except Exception as e:
            print(f"{e}")
    else:
        return None
```

Full example: dns resource list

Find Zone

The find_zone function searches for and returns a DNS zone by its name using a given connection object.

```
def find_zone(conn, name_or_id):
    print(f"Find Zone: {name_or_id}")

    zone = conn.dns.find_zone(name_or_id)

    if zone:
        print(zone)
        return zone
    else:
        print("Zone not found.")
        return None
```

Full example: dns resource list

Find Recordset

The find_recordset function searches for a DNS recordset with a specific name and type within a given zone. If multiple recordsets with the same name exist, the record type can be specified to find the exact match.

```
def find_recordset(conn, name_or_id, recordset_name, recordset_type=None):
    print(f"Find Recordset: {recordset_name} in Zone: {name_or_id}")

    zone = conn.dns.find_zone(name_or_id)

```

(continues on next page)

(continued from previous page)

```
    if not zone:
        print("Zone not found.")
        return None

    zone_id = zone.id

    try:
        if recordset_type:
            recordset = conn.dns.find_recordset(
                zone_id, recordset_name, type=recordset_type
            )
        else:
            recordset = conn.dns.find_recordset(zone_id, recordset_name)

        if recordset:
            print(recordset)
            return recordset
        else:
            print("Recordset not found in Zone.")
            return None

    except Exception as e:
        print(f"{e}")
        return None
```

Full example: dns resource list

Using OpenStack Identity

Before working with the Identity service, youll need to create a connection to your OpenStack cloud by following the *Connect* user guide. This will provide you with the `conn` variable used in the examples below.

The OpenStack Identity service is the default identity management system for OpenStack. The Identity service authentication process confirms the identity of a user and an incoming request by validating a set of credentials that the user supplies. Initially, these credentials are a user name and password or a user name and API key. When the Identity service validates user credentials, it issues an authentication token that the user provides in subsequent requests. An authentication token is an alpha-numeric text string that enables access to OpenStack APIs and resources. A token may be revoked at any time and is valid for a finite duration.

List Users

A **user** is a digital representation of a person, system, or service that uses OpenStack cloud services. The Identity service validates that incoming requests are made by the user who claims to be making the call. Users have a login and can access resources by using assigned tokens. Users can be directly assigned to a particular project and behave as if they are contained in that project.

```
def list_users(conn):
    print("List Users:")
```

(continues on next page)

(continued from previous page)

```
    for user in conn.identity.users():
        print(user)
```

Full example: identity resource list

List Credentials

Credentials are data that confirms the identity of the user. For example, user name and password, user name and API key, or an authentication token that the Identity service provides.

```
def list_credentials(conn):
    print("List Credentials:")

    for credential in conn.identity.credentials():
        print(credential)
```

Full example: identity resource list

List Projects

A **project** is a container that groups or isolates resources or identity objects.

```
def list_projects(conn):
    print("List Projects:")

    for project in conn.identity.projects():
        print(project)
```

Full example: identity resource list

List Domains

A **domain** is an Identity service API v3 entity and represents a collection of projects and users that defines administrative boundaries for the management of Identity entities. Users can be granted the administrator role for a domain. A domain administrator can create projects, users, and groups in a domain and assign roles to users and groups in a domain.

```
def list_domains(conn):
    print("List Domains:")

    for domain in conn.identity.domains():
        print(domain)
```

Full example: identity resource list

List Groups

A **group** is an Identity service API v3 entity and represents a collection of users that are owned by a domain. A group role granted to a domain or project applies to all users in the group. Adding users to, or removing users from, a group respectively grants, or revokes, their role and authentication to the associated domain or project.

```
def list_groups(conn):
    print("List Groups:")

    for group in conn.identity.groups():
        print(group)
```

Full example: identity resource list

List Services

A **service** is an OpenStack service, such as Compute, Object Storage, or Image service, that provides one or more endpoints through which users can access resources and perform operations.

```
def list_services(conn):
    print("List Services:")

    for service in conn.identity.services():
        print(service)
```

Full example: identity resource list

List Endpoints

An **endpoint** is a network-accessible address, usually a URL, through which you can access a service.

```
def list_endpoints(conn):
    print("List Endpoints:")

    for endpoint in conn.identity.endpoints():
        print(endpoint)
```

Full example: identity resource list

List Regions

A **region** is an Identity service API v3 entity and represents a general division in an OpenStack deployment. You can associate zero or more sub-regions with a region to make a tree-like structured hierarchy.

```
def list_regions(conn):
    print("List Regions:")

    for region in conn.identity.regions():
        print(region)
```

Full example: identity resource list

Using OpenStack Image

Before working with the Image service, youll need to create a connection to your OpenStack cloud by following the *Connect* user guide. This will provide you with the `conn` variable used in the examples below.

The primary resource of the Image service is the image.

List Images

An **image** is a collection of files for a specific operating system that you use to create or rebuild a server. OpenStack provides pre-built images. You can also create custom images, or snapshots, from servers that you have launched. Images come in different formats and are sometimes called virtual machine images.

```
def list_images(conn):
    print("List Images:")

    for image in conn.image.images():
        print(image)
```

Full example: image resource list

Create Image

Create an image by uploading its data and setting its attributes.

```
def upload_image(conn):
    print("Upload Image:")

    # Load fake image data for the example.
    data = 'This is fake image data.'

    # Build the image attributes and upload the image.
    image_attrs = {
        'name': EXAMPLE_IMAGE_NAME,
        'data': data,
        'disk_format': 'raw',
        'container_format': 'bare',
        'visibility': 'public',
    }
    conn.image.upload_image(**image_attrs)
```

Full example: image resource create

Create Image via interoperable image import process

Create an image then use interoperable image import process to download data from a web URL.

For more information about the image import process, please check interoperable image import

```
def import_image(conn):
    print("Import Image:")

    # Url where glance can download the image
    uri = (
        'https://download.cirros-cloud.net/0.4.0/cirros-0.4.0-x86_64-disk.img'
    )

    # Build the image attributes and import the image.
    image_attrs = {
```

(continues on next page)

(continued from previous page)

```
        'name': EXAMPLE_IMAGE_NAME,
        'disk_format': 'qcow2',
        'container_format': 'bare',
        'visibility': 'public',
    }
    image = conn.image.create_image(**image_attrs)
    conn.image.import_image(image, method="web-download", uri=uri)
```

Full example: image resource import

Downloading an Image with stream=True

As images are often very large pieces of data, storing their entire contents in the memory of your application can be less than desirable. A more efficient method may be to iterate over a stream of the response data.

By choosing to stream the response content, you determine the `chunk_size` that is appropriate for your needs, meaning only that many bytes of data are read for each iteration of the loop until all data has been consumed. See `requests.Response.iter_content()` for more information.

When you choose to stream an image download, openstacksdk is no longer able to compute the checksum of the response data for you. This example shows how you might do that yourself, in a very similar manner to how the library calculates checksums for non-streamed responses.

```
def download_image_stream(conn):
    print("Download Image via streaming:")

    # Find the image you would like to download.
    image = conn.image.find_image("myimage")

    # As the actual download now takes place outside of the library
    # and in your own code, you are now responsible for checking
    # the integrity of the data. Create an MD5 has to be computed
    # after all of the data has been consumed.
    md5 = hashlib.md5()

    with open("myimage.qcow2", "wb") as local_image:
        response = conn.image.download_image(image, stream=True)

        # Read only 1 MiB of memory at a time until
        # all of the image data has been consumed.
        for chunk in response.iter_content(chunk_size=1024 * 1024):
            # With each chunk, add it to the hash to be computed.
            md5.update(chunk)

            local_image.write(chunk)

        # Now that you've consumed all of the data the response gave you,
        # ensure that the checksums of what the server offered and
        # what you downloaded are the same.
        if response.headers["Content-MD5"] != md5.hexdigest():
```

(continues on next page)

(continued from previous page)

```
            raise Exception("Checksum mismatch in downloaded content")
```

Downloading an Image with stream=False

If you wish to download an images contents all at once and to memory, simply set `stream=False`, which is the default.

```
def download_image(conn):
    print("Download Image:")

    # Find the image you would like to download.
    image = conn.image.find_image("myimage")

    with open("myimage.qcow2", "w") as local_image:
        response = conn.image.download_image(image)

        # Response will contain the entire contents of the Image.
        local_image.write(response)
```

Full example: image resource download

Delete Image

Delete an image.

```
def delete_image(conn):
    print("Delete Image:")

    example_image = conn.image.find_image(EXAMPLE_IMAGE_NAME)

    conn.image.delete_image(example_image, ignore_missing=False)
```

Full example: image resource delete

Using OpenStack Key Manager

Before working with the Key Manager service, youll need to create a connection to your OpenStack cloud by following the *Connect* user guide. This will provide you with the `conn` variable used in the examples below.

Table of Contents

Note

Some interactions with the Key Manager service differ from that of other services in that resources do not have a proper `id` parameter, which is necessary to make some calls. Instead, resources have a separately named id attribute, e.g., the Secret resource has `secret_id`.

The examples below outline when to pass in those id values.

Create a Secret

The Key Manager service allows you to create new secrets by passing the attributes of the *Secret* to the `create_secret()` method.

```
def create_secret(conn):
    print("Create a secret:")

    conn.key_manager.create_secret(
        name="My public key",
        secret_type="public",
        expiration="2020-02-28T23:59:59",
        payload="ssh rsa...",
        payload_content_type="text/plain",
    )
```

List Secrets

Once you have stored some secrets, they are available for you to list via the `secrets()` method. This method returns a generator, which yields each *Secret*.

```
def list_secrets(conn):
    print("List Secrets:")

    for secret in conn.key_manager.secrets():
        print(secret)
```

The `secrets()` method can also make more advanced queries to limit the secrets that are returned.

```
def list_secrets_query(conn):
    print("List Secrets:")

    for secret in conn.key_manager.secrets(
        secret_type="symmetric", expiration="gte:2020-01-01T00:00:00"
    ):
        print(secret)
```

Get Secret Payload

Once you have received a *Secret*, you can obtain the payload for it by passing the secrets id value to the `secrets()` method. Use the `secret_id` attribute when making this request.

```
def get_secret_payload(conn):
    print("Get a secret's payload:")

    # Assuming you have an object `s` which you perhaps received from
    # a conn.key_manager.secrets() call...
    secret = conn.key_manager.get_secret(s.secret_id)
    print(secret.payload)
```

Using OpenStack Message

Before working with the Message service, youll need to create a connection to your OpenStack cloud by following the *Connect* user guide. This will provide you with the `conn` variable used in the examples below.

Using OpenStack Network

Before working with the Network service, youll need to create a connection to your OpenStack cloud by following the *Connect* user guide. This will provide you with the `conn` variable used in the examples below.

Table of Contents

The primary resource of the Network service is the network.

List Networks

A **network** is an isolated Layer 2 networking segment. There are two types of networks, project and provider networks. Project networks are fully isolated and are not shared with other projects. Provider networks map to existing physical networks in the data center and provide external network access for servers. Only an OpenStack administrator can create provider networks. Networks can be connected via routers.

```
def list_networks(conn):
    print("List Networks:")
```

(continues on next page)

(continued from previous page)

```
    for network in conn.network.networks():
        print(network)
```

Full example: network resource list

List Subnets

A **subnet** is a block of IP addresses and associated configuration state. Subnets are used to allocate IP addresses when new ports are created on a network.

```
def list_subnets(conn):
    print("List Subnets:")

    for subnet in conn.network.subnets():
        print(subnet)
```

Full example: network resource list

List Ports

A **port** is a connection point for attaching a single device, such as the NIC of a server, to a network. The port also describes the associated network configuration, such as the MAC and IP addresses to be used on that port.

```
def list_ports(conn):
    print("List Ports:")

    for port in conn.network.ports():
        print(port)
```

Full example: network resource list

List Security Groups

A **security group** acts as a virtual firewall for servers. It is a container for security group rules which specify the type of network traffic and direction that is allowed to pass through a port.

```
def list_security_groups(conn):
    print("List Security Groups:")

    for port in conn.network.security_groups():
        print(port)
```

Full example: network resource list

List Routers

A **router** is a logical component that forwards data packets between networks. It also provides Layer 3 and NAT forwarding to provide external network access for servers on project networks.

```
def list_routers(conn):
    print("List Routers:")

    for router in conn.network.routers():
        print(router)
```

Full example: network resource list

List Network Agents

A **network agent** is a plugin that handles various tasks used to implement virtual networks. These agents include neutron-dhcp-agent, neutron-l3-agent, neutron-metering-agent, and neutron-lbaas-agent, among others.

```
def list_network_agents(conn):
    print("List Network Agents:")

    for agent in conn.network.agents():
        print(agent)
```

Full example: network resource list

Create Network

Create a project network and subnet. This network can be used when creating a server and allows the server to communicate with others servers on the same project network.

```
def create_network(conn):
    print("Create Network:")

    example_network = conn.network.create_network(
        name='openstacksdk-example-project-network'
    )

    print(example_network)

    example_subnet = conn.network.create_subnet(
        name='openstacksdk-example-project-subnet',
        network_id=example_network.id,
        ip_version='4',
        cidr='10.0.2.0/24',
        gateway_ip='10.0.2.1',
    )

    print(example_subnet)
```

Full example: network resource create

Open a Port

When creating a security group for a network, you will need to open certain ports to allow communication via them. For example, you may need to enable HTTPS access on port 443.

```
def open_port(conn):
    print("Open a port:")

    example_sec_group = conn.network.create_security_group(
        name='openstacksdk-example-security-group'
    )

    print(example_sec_group)

    example_rule = conn.network.create_security_group_rule(
        security_group_id=example_sec_group.id,
        direction='ingress',
        remote_ip_prefix='0.0.0.0/0',
        protocol='tcp',
        port_range_max='443',
        port_range_min='443',
        ethertype='IPv4',
    )

    print(example_rule)
```

Full example: network security group create

Accept Pings

In order to ping a machine on your network within a security group, you will need to create a rule to allow inbound ICMP packets.

```
def allow_ping(conn):
    print("Allow pings:")

    example_sec_group = conn.network.create_security_group(
        name='openstacksdk-example-security-group2'
    )

    print(example_sec_group)

    example_rule = conn.network.create_security_group_rule(
        security_group_id=example_sec_group.id,
        direction='ingress',
        remote_ip_prefix='0.0.0.0/0',
        protocol='icmp',
        port_range_max=None,
        port_range_min=None,
        ethertype='IPv4',
    )
```

(continues on next page)

(continued from previous page)

```
    print(example_rule)
```

Full example: network security group create

Delete Network

Delete a project network and its subnets.

```
def delete_network(conn):
    print("Delete Network:")

    example_network = conn.network.find_network(
        'openstacksdk-example-project-network'
    )

    for example_subnet in example_network.subnet_ids:
        conn.network.delete_subnet(example_subnet, ignore_missing=False)
    conn.network.delete_network(example_network, ignore_missing=False)
```

Full example: network resource delete

Using OpenStack Object Store

Before working with the Object Store service, youll need to create a connection to your OpenStack cloud by following the *Connect* user guide. This will provide you with the `conn` variable used in the examples below.

Table of Contents

The primary resources of the Object Store service are containers and objects.

Working with Containers

Listing Containers

To list existing containers, use the `containers()` method.

```
>>> for cont in conn.object_store.containers():
...     print cont
...
openstack.object_store.v1.container.Container: {u'count': 5,
u'bytes': 500, u'name': u'my container'}
openstack.object_store.v1.container.Container: {u'count': 0,
u'bytes': 0, u'name': u'empty container'}
openstack.object_store.v1.container.Container: {u'count': 100,
u'bytes': 1000000, u'name': u'another container'}
```

The `containers` method returns a generator which yields *Container* objects. It handles pagination for you, which can be adjusted via the `limit` argument. By default, the `containers` method will yield as many containers as the service will return, and it will continue requesting until it receives no more.

```
>>> for cont in conn.object_store.containers(limit=500):
...     print(cont)
...
<500 Containers>
... another request transparently made to the Object Store service
<500 more Containers>
...
```

Creating Containers

To create a container, use the `create_container()` method.

```
>>> cont = conn.object_store.create_container(name="new container")
>>> cont
openstack.object_store.v1.container.Container: {'name': u'new container'}
```

Working with Container Metadata

To get the metadata for a container, use the `get_container_metadata()` method. This method either takes the name of a container, or a *Container* object, and it returns a *Container* object with all of its metadata attributes set.

```
>>> cont = conn.object_store.get_container_metadata("new container")
openstack.object_store.v1.container.Container: {'content-length': '0',
'x-container-object-count': '0', 'name': u'new container',
'accept-ranges': 'bytes',
'x-trans-id': 'tx22c5de63466e4c05bb104-0054740c39',
'date': 'Tue, 25 Nov 2014 04:57:29 GMT',
'x-timestamp': '1416889793.23520', 'x-container-read': '.r:mysite.com',
'x-container-bytes-used': '0', 'content-type': 'text/plain; charset=utf-8'}
```

To set the metadata for a container, use the `set_container_metadata()` method. This method takes

a *Container* object. For example, to grant another user write access to this container, you can call *set_container_metadata* passing it the *Container* to update and keyward argument key/value pairs representing the metadata name and value to set.

```
>>> acl = "big_project:another_user"
>>> conn.object_store.set_container_metadata(cont, write_ACL=acl)
openstack.object_store.v1.container.Container: {'content-length': '0',
'x-container-object-count': '0',
'name': u'my new container', 'accept-ranges': 'bytes',
'x-trans-id': 'txc3ee751f971d41de9e9f4-0054740ec1',
'date': 'Tue, 25 Nov 2014 05:08:17 GMT',
'x-timestamp': '1416889793.23520', 'x-container-read': '.r:mysite.com',
'x-container-bytes-used': '0', 'content-type': 'text/plain; charset=utf-8',
'x-container-write': 'big_project:another_user'}
```

Working with Objects

Objects are held in containers. From an API standpoint, you work with them using similarly named methods, typically with an additional argument to specify their container.

Listing Objects

To list the objects that exist in a container, use the `objects()` method.

If you have a *Container* object, you can pass it to `objects`.

```
>>> print cont.name
pictures
>>> for obj in conn.object_store.objects(cont):
...     print obj
...
openstack.object_store.v1.container.Object:
{u'hash': u'0522d4ccdf9956badcb15c4087a0c4cb',
u'name': u'pictures/selfie.jpg', u'bytes': 15744,
'last-modified': u'2014-10-31T06:33:36.618640',
u'last_modified': u'2014-10-31T06:33:36.618640',
u'content_type': u'image/jpeg', 'container': u'pictures',
'content-type': u'image/jpeg'}
...
```

Similar to the `containers()` method, `objects` returns a generator which yields *Object* objects stored in the container. It also handles pagination for you, which you can adjust with the `limit` parameter, otherwise making each request for the maximum that your Object Store will return.

If you have the name of a container instead of an object, you can also pass that to the `objects` method.

```
>>> for obj in conn.object_store.objects("pictures".decode("utf8"),
                                         limit=100):
...     print obj
...
<100 Objects>
```

(continues on next page)

(continued from previous page)

```
... another request transparently made to the Object Store service
<100 more Objects>
```

Getting Object Data

Once you have an *Object*, you get the data stored inside of it with the `get_object_data()` method.

```
>>> print ob.name
message.txt
>>> data = conn.object_store.get_object_data(ob)
>>> print data
Hello, world!
```

Additionally, if you want to save the object to disk, the `download_object()` convenience method takes an *Object* and a `path` to write the contents to.

```
>>> conn.object_store.download_object(ob, "the_message.txt")
```

Uploading Objects

Once you have data youd like to store in the Object Store service, you use the `upload_object()` method. This method takes the `data` to be stored, along with at least an object `name` and the `container` it is to be stored in.

```
>>> hello = conn.object_store.upload_object(container="messages",
                                            name="helloworld.txt",
                                            data="Hello, world!")
>>> print hello
openstack.object_store.v1.container.Object: {'content-length': '0',
'container': u'messages', 'name': u'helloworld.txt',
'last-modified': 'Tue, 25 Nov 2014 17:39:29 GMT',
'etag': '5eb63bbbe01eeed093cb22bb8f5acdc3',
'x-trans-id': 'tx3035d41b03334aeaaf3dd-005474bed0',
'date': 'Tue, 25 Nov 2014 17:39:28 GMT',
'content-type': 'text/html; charset=UTF-8'}
```

Working with Object Metadata

Working with metadata on objects is identical to how its done with containers. You use the `get_object_metadata()` and `set_object_metadata()` methods.

The metadata attributes to be set can be found on the *Object* object.

```
>>> secret.delete_after = 300
>>> secret = conn.object_store.set_object_metadata(secret)
```

We set the `delete_after` value to 500 seconds, causing the object to be deleted in 300 seconds, or five minutes. That attribute corresponds to the `X-Delete-After` header value, which you can see is returned when we retrieve the updated metadata.

```
>>> conn.object_store.get_object_metadata(ob)
openstack.object_store.v1.container.Object: {'content-length': '11',
'container': u'Secret Container',
'name': u'selfdestruct.txt', 'x-delete-after': 300,
'accept-ranges': 'bytes', 'last-modified': 'Tue, 25 Nov 2014 17:50:45 GMT',
'etag': '5eb63bbbe01eeed093cb22bb8f5acdc3',
'x-timestamp': '1416937844.36805',
'x-trans-id': 'tx5c3fd94adf7c4e1b8f334-005474c17b',
'date': 'Tue, 25 Nov 2014 17:50:51 GMT', 'content-type': 'text/plain'}
```

Using OpenStack Orchestration

Before working with the Orchestration service, youll need to create a connection to your OpenStack cloud by following the *Connect* user guide. This will provide you with the `conn` variable used in the examples below.

Using OpenStack Shared File Systems

Before working with the Shared File System service, youll need to create a connection to your OpenStack cloud by following the *Connect* user guide. This will provide you with the `conn` variable used in the examples below.

Table of Contents

- *Delete Share Metadata*
- *Manage Share*
- *Unmanage Share*

List Availability Zones

A Shared File System service **availability zone** is a failure domain for your shared file systems. You may create a shared file system (referred to simply as **shares**) in a given availability zone, and create replicas of the share in other availability zones.

```
def list_availability_zones(conn):
    print("List Shared File System Availability Zones:")
    for az in conn.share.availability_zones():
        print(az)
```

Share Instances

Administrators can list, show information for, explicitly set the state of, and force-delete share instances.

```
def share_instances(conn, **query):
    print('List all share instances:')
    for si in conn.share.share_instances(**query):
        print(si)
```

Get Share Instance

Shows details for a single share instance.

```
def get_share_instance(conn, share_instance_id):
    print('Get share instance with given Id:')
    share_instance = conn.share.get_share_instance(share_instance_id)
    print(share_instance)
```

Reset Share Instance Status

Explicitly updates the state of a share instance.

```
def reset_share_instance_status(conn, share_instance_id, status):
    print(
        'Reset the status of the share instance with the given '
        'share_instance_id to the given status'
    )
    conn.share.reset_share_instance_status(share_instance_id, status)
```

Delete Share Instance

Force-deletes a share instance.

```
def delete_share_instance(conn, share_instance_id):
    print('Force-delete the share instance with the given share_instance_id')
    conn.share.delete_share_instance(share_instance_id)
```

Resize Share

Shared File System shares can be resized (extended or shrunk) to a given size. For details on resizing shares, refer to the Manila docs.

```
def resize_share(conn, share_id, share_size):
    # Be explicit about not wanting to use force if the share
    # will be extended.
    use_force = False
    print('Resize the share to the given size:')
    conn.share.resize_share(share_id, share_size, use_force)
```

```
def resize_shares_without_shrink(conn, min_size):
    # Sometimes, extending shares without shrinking
    # them (effectively setting a min size) is desirable.

    # Get list of shares from the connection.
    shares = conn.share.shares()

    # Loop over the shares:
    for share in shares:
        # Extend shares smaller than min_size to min_size,
        # but don't shrink shares larger than min_size.
        conn.share.resize_share(share.id, min_size, no_shrink=True)
```

List Share Group Snapshots

A share group snapshot is a point-in-time, read-only copy of the data that is contained in a share group. You can list all share group snapshots

```
def list_share_group_snapshots(conn, **query):
    print("List all share group snapshots:")
    share_group_snapshots = conn.share.share_group_snapshots(**query)
    for share_group_snapshot in share_group_snapshots:
        print(share_group_snapshot)
```

Get Share Group Snapshot

Show share group snapshot details

```
def get_share_group_snapshot(conn, group_snapshot_id):
    print("Show share group snapshot with given Id:")
```

(continues on next page)

(continued from previous page)

```
    share_group_snapshot = conn.share.get_share_group_snapshots(
        group_snapshot_id
    )
    print(share_group_snapshot)
```

List Share Group Snapshot Members

Lists all share group snapshots members.

```
def share_group_snapshot_members(conn, group_snapshot_id):
    print("Show share group snapshot members with given Id:")
    members = conn.share.share_group_snapshot_members(group_snapshot_id)
    for member in members:
        print(member)
```

Create Share Group Snapshot

Creates a snapshot from a share group.

```
def create_share_group_snapshot(conn, share_group_id, **attrs):
    print("Creating a share group snapshot from given attributes:")
    share_group_snapshot = conn.share.create_share_group_snapshot(
        share_group_id, **attrs
    )
    print(share_group_snapshot)
```

Reset Share Group Snapshot

Reset share group snapshot state.

```
def reset_share_group_snapshot_status(conn, group_snapshot_id, status):
    print("Reseting the share group snapshot status:")
    conn.share.reset_share_group_snapshot_status(group_snapshot_id, status)
```

Update Share Group Snapshot

Updates a share group snapshot.

```
def update_share_group_snapshot(conn, group_snapshot_id, **attrs):
    print("Updating a share group snapshot with given Id:")
    share_group_snapshot = conn.share.update_share_group_snapshot(
        group_snapshot_id, **attrs
    )
    print(share_group_snapshot)
```

Delete Share Group Snapshot

Deletes a share group snapshot.

```
def delete_share_group_snapshot(conn, group_snapshot_id):
    print("Deleting a share group snapshot with given Id:")
    conn.share.delete_share_group_snapshot(group_snapshot_id)
```

List Share Metadata

Lists all metadata for a given share.

```
def list_share_metadata(conn, share_id):
    # Method returns the entire share with the metadata inside it.
    returned_share = conn.get_share_metadata(share_id)

    # Access metadata of share
    metadata = returned_share['metadata']

    print("List All Share Metadata:")
    for meta_key in metadata:
        print(f"{meta_key}={metadata[meta_key]}")
```

Get Share Metadata Item

Retrieves a specific metadata item from a shares metadata by its key.

```
def get_share_metadata_item(conn, share_id, key):
    # Method returns the entire share with the metadata inside it.
    returned_share = conn.get_share_metadata_item(share_id, key)

    # Access metadata of share
    metadata = returned_share['metadata']

    print("Get share metadata item given item key and share id:")
    print(metadata[key])
```

Create Share Metadata

Creates share metadata.

```
def create_share_metadata(conn, share_id, metadata):
    # Method returns the entire share with the metadata inside it.
    created_share = conn.create_share_metadata(share_id, metadata)

    # Access metadata of share
    metadata = created_share['metadata']

    print("Metadata created for given share:")
    print(metadata)
```

Update Share Metadata

Updates metadata of a given share.

```
def update_share_metadata(conn, share_id, metadata):
    # Method returns the entire share with the metadata inside it.
    updated_share = conn.update_share_metadata(share_id, metadata, True)

    # Access metadata of share
    metadata = updated_share['metadata']

    print("Updated metadata for given share:")
    print(metadata)
```

Delete Share Metadata

Deletes a specific metadata item from a shares metadata by its key. Can specify multiple keys to be deleted.

```
def delete_share_metadata(conn, share_id, keys):
    # Method doesn't return anything.
    conn.delete_share_metadata(share_id, keys)
```

Manage Share

Manage a share with Manila.

```
def manage_share(conn, protocol, export_path, service_host, **params):
    # Manage a share with the given protocol, export path, service host, and
    # optional additional parameters
    managed_share = conn.share.manage_share(
        protocol, export_path, service_host, **params
    )

    # Can get the ID of the share, which is now being managed with Manila
    managed_share_id = managed_share.id
    print("The ID of the share which was managed: %s", managed_share_id)
```

Unmanage Share

Unmanage a share from Manila.

```
def unmanage_share(conn, share_id):
    # Unmanage the share with the given share ID
    conn.share.unmanage_share(share_id)

    try:
        # Getting the share will raise an exception as it has been unmanaged
        conn.share.get_share(share_id)
    except Exception:
        pass
```

2.1.2 Testing

The SDK provides a number of utilities to help you test your applications.

Testing applications using OpenStack SDK

Fakes

The *fakes* module exists to help application developers using the OpenStack SDK to unit test their applications. It provides a number of helper utilities to generate fake *Resource* and *Proxy* instances. These fakes do not require an established connection and allow you to validate that your application using valid attributes and methods for both *Resource* and *Proxy* instances.

openstack.test.fakes.**generate_fake_resource**(*resource_type*, ***attrs*)

Generate a fake resource

Parameters

- **resource_type** (*type*) Object class
- **attrs** (*dict*) Optional attributes to be set on resource

Example usage:

```
>>> from openstack.compute.v2 import server
>>> from openstack.test import fakes
>>> fakes.generate_fake_resource(server.Server)
openstack.compute.v2.server.Server(...)
```

Parameters

- **resource_type** (*type*) Object class
- **attrs** (*dict*) Optional attributes to be set on resource

Returns

Instance of `resource_type` class populated with fake values of expected types

Raises

NotImplementedError If a resource attribute specifies a `type` or `list_type` that cannot be automatically generated

openstack.test.fakes.**generate_fake_resources**(*resource_type*, *count=1*, *attrs=None*)

Generate a given number of fake resource entities

Parameters

- **resource_type** (*type*) Object class
- **count** (*int*) Number of objects to return
- **attrs** (*dict*) Attribute values to set into each instance

Example usage:

```
>>> from openstack.compute.v2 import server
>>> from openstack.test import fakes
>>> fakes.generate_fake_resources(server.Server, count=3)
<generator object generate_fake_resources at 0x7f075dc65040>
```

Parameters

- **resource_type** (*type*) Object class
- **count** (*int*) Number of objects to return
- **attrs** (*dict*) Attribute values to set into each instance

Returns

Generator of `resource_type` class instances populated with fake values of expected types.

`openstack.test.fakes.`**generate_fake_proxy**(*service*, *api_version=None*)

Generate a fake proxy for the given service type

Example usage:

```
>>> import functools
>>> from openstack.compute import compute_service
>>> from openstack.compute.v2 import server
>>> from openstack.test import fakes
>>> # create the fake proxy
>>> fake_compute_proxy = fakes.generate_fake_proxy(
...     compute_service.ComputeService,
... )
>>> # configure return values for various proxy APIs
>>> # note that this will generate new fake resources on each invocation
>>> fake_compute_proxy.get_server.side_effect = functools.partial(
...     fakes.generate_fake_resource,
...     server.Server,
... )
>>> fake_compute_proxy.servers.side_effect = functools.partial(
...     fakes.generate_fake_resources,
...     server.Server,
... )
>>> fake_compute_proxy.servers()
<generator object generate_fake_resources at 0x7f92768dc040>
>>> fake_compute_proxy.serverssss()
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "/usr/lib64/python3.11/unittest/mock.py", line 653, in __getattr__
    raise AttributeError("Mock object has no attribute %r" % name)
AttributeError: Mock object has no attribute 'serverssss'. Did you mean:
↪'server_ips'?
```

Parameters

- **service** (*ServiceDescription*) The service to generate the fake proxy for.
- **api_version** (*int or None*) The API version to generate the fake proxy for. This should be a major version must be supported by openstacksdk, as specified in the `supported_versions` attribute of the provided `service`. This is only required if openstacksdk supports multiple API versions for the given service.

Raises

ValueError if the `service` is not a valid *ServiceDescription* or if `api_version` is not supported

Returns

An autospecced mock of the *Proxy* implementation for the specified service type and API version

2.1.3 API Documentation

Service APIs are exposed through a two-layered approach. The classes exposed through our *Connection* interface are the place to start if youre an application developer consuming an OpenStack cloud. The *Resource* interface is the layer upon which the *Connection* is built, with *Connection* methods accepting and returning *Resource* objects.

The Cloud Abstraction layer has a data model.

Data Model

openstacksdk has a very strict policy on not breaking backwards compatibility ever. However, with the data structures returned from OpenStack, there are places where the resource structures from OpenStack are returned to the user somewhat directly, leaving an openstacksdk user open to changes/differences in result content.

To combat that, openstacksdk normalizes the return structure from OpenStack in many places, and the results of that normalization are listed below. Where openstacksdk performs normalization, a user can count on any fields declared in the docs as being completely safe to use - they are as much a part of openstacksdks API contract as any other Python method.

Some OpenStack objects allow for arbitrary attributes at the root of the object. openstacksdk will pass those through so as not to break anyone who may be counting on them, but as they are arbitrary openstacksdk can make no guarantees as to their existence. As part of normalization, openstacksdk will put any attribute from an OpenStack resource that is not in its data model contract into an attribute called properties. The contents of properties are defined to be an arbitrary collection of key value pairs with no promises as to any particular key ever existing.

If a user passes `strict=True` to the openstacksdk constructor, openstacksdk will not pass through arbitrary objects to the root of the resource, and will instead only put them in the properties dict. If a user is worried about accidentally writing code that depends on an attribute that is not part of the API contract, this can be a useful tool. Keep in mind all data can still be accessed via the properties dict, but any code touching anything in the properties dict should be aware that the keys found there are highly user/cloud specific. Any key that is transformed as part of the openstacksdk data model contract will not wind up with an entry in properties - only keys that are unknown.

The `location` field

A Location defines where a resource lives. It includes a cloud name and a region name, an availability zone as well as information about the project that owns the resource.

The project information may contain a project ID, or a combination of one or more of a project name with a domain name or ID. If a project ID is present, it should be considered correct.

Some resources do not carry ownership information with them. For those, the project information will be filled in from the project the user currently has a token for.

Some resources do not have information about availability zones, or may exist region wide. Those resources will have None as their availability zone.

```
Location = dict(
    cloud=str(),
    region_name=str(),
    zone=str() or None,
    project=dict(
        id=str() or None,
        name=str() or None,
        domain_id=str() or None,
        domain_name=str() or None,
    )
)
```

Connection Interface

A *Connection* instance maintains your cloud config, session and authentication information providing you with a set of higher-level interfaces to work with OpenStack services.

Connection

The `Connection` class is the primary interface to the Python SDK. It maintains a context for a connection to a region of a cloud provider. The `Connection` has an attribute to access each OpenStack service.

At a minimum, the `Connection` class needs to be created with a config or the parameters to build one.

While the overall system is very flexible, there are four main use cases for different ways to create a `Connection`.

- Using config settings and keyword arguments as described in *Configuring OpenStack SDK Applications*
- Using only keyword arguments passed to the constructor ignoring config files and environment variables.
- Using an existing authenticated *keystoneauth1.session.Session*, such as might exist inside of an OpenStack service operational context.
- Using an existing `CloudRegion`.

Creating the Connection

Using config settings

For users who want to create a `Connection` making use of named clouds in `clouds.yaml` files, `OS_` environment variables and python keyword arguments, the `openstack.connect()` factory function is the recommended way to go:

```
import openstack

conn = openstack.connect(cloud='example', region_name='earth1')
```

If the application in question is a command line application that should also accept command line arguments, an *argparse.Namespace* can be passed to `openstack.connect()` that will have relevant arguments added to it and then subsequently consumed by the constructor:

```
import argparse
import openstack

options = argparse.ArgumentParser(description='Awesome OpenStack App')
conn = openstack.connect(options=options)
```

Using only keyword arguments

If the application wants to avoid loading any settings from `clouds.yaml` or environment variables, use the *Connection* constructor directly. As long as the `cloud` argument is omitted or `None`, the *Connection* constructor will not load settings from files or the environment.

Note

This is a different default behavior than the `connect()` factory function. In `connect()` if `cloud` is omitted or `None`, a default cloud will be loaded, defaulting to the `envvars` cloud if it exists.

```
from openstack import connection

conn = connection.Connection(
    region_name='example-region',
    auth={
        'auth_url': 'https://auth.example.com',
        'username': 'amazing-user',
        'password': 'super-secret-password',
        'project_id': '33aa1afc-03fe-43b8-8201-4e0d3b4b8ab5',
        'user_domain_id': '054abd68-9ad9-418b-96d3-3437bb376703',
    },
    compute_api_version='2',
    identity_interface='internal',
)
```

Per-service settings as needed by *keystoneauth1.adapter.Adapter* such as `api_version`, `service_name`, and `interface` can be set, as seen above, by prefixing them with the official `service-type` name of the service. `region_name` is a setting for the entire *CloudRegion* and cannot be set per service.

From existing authenticated Session

For applications that already have an authenticated Session, simply passing it to the *Connection* constructor is all that is needed:

```
from openstack import connection

conn = connection.Connection(
    session=session,
    region_name='example-region',
```

(continues on next page)

(continued from previous page)

```
    compute_api_version='2',
    identity_interface='internal',
)
```

From oslo.conf CONF object

For applications that have an oslo.config `CONF` object that has been populated with `keystoneauth1.loading.register_adapter_conf_options` in groups named by the OpenStack services project name, it is possible to construct a Connection with the `CONF` object and an authenticated Session.

Note

This is primarily intended for use by OpenStack services to talk amongst themselves.

```
from keystoneauth1 import loading as ks_loading
from oslo_config import cfg
from openstack import connection

CONF = cfg.CONF

group = cfg.OptGroup('neutron')
ks_loading.register_session_conf_options(CONF, group)
ks_loading.register_auth_conf_options(CONF, group)
ks_loading.register_adapter_conf_options(CONF, group)

CONF()

auth = ks_loading.load_auth_from_conf_options(CONF, 'neutron')
sess = ks_loading.load_session_from_conf_options(CONF, 'neutron', auth=auth)

conn = connection.Connection(
    session=sess,
    oslo_conf=CONF,
)
```

This can then be used with an appropriate configuration file.

```
[neutron]
region_name = RegionOne
auth_strategy = keystone
project_domain_name = Default
project_name = service
user_domain_name = Default
password = password
username = neutron
auth_url = http://10.0.110.85/identity
auth_type = password
service_metadata_proxy = True
```

(continues on next page)

(continued from previous page)

```
default_floating_pool = public
```

You may also wish to configure a service user. As discussed in the Keystone documentation, service users are users with specific roles that identify the user as a service. The use of service users can avoid issues caused by the expiration of the original users token during long running operations, as a fresh token issued for the service user will always accompany the users token, which may have expired.

```
from keystoneauth1 import loading as ks_loading
from keystoneauth1 import service_token
from oslo_config import cfg
import openstack
from openstack import connection

CONF = cfg.CONF

neutron_group = cfg.OptGroup('neutron')
ks_loading.register_session_conf_options(CONF, neutron_group)
ks_loading.register_auth_conf_options(CONF, neutron_group)
ks_loading.register_adapter_conf_options(CONF, neutron_group)

service_group = cfg.OptGroup('service_user')
ks_loading.register_session_conf_options(CONF, service_group)
ks_loading.register_auth_conf_options(CONF, service_group)

CONF()
user_auth = ks_loading.load_auth_from_conf_options(CONF, 'neutron')
service_auth = ks_loading.load_auth_from_conf_options(CONF, 'service_user')
auth = service_token.ServiceTokenAuthWrapper(user_auth, service_auth)

sess = ks_loading.load_session_from_conf_options(CONF, 'neutron', auth=auth)

conn = connection.Connection(
    session=sess,
    oslo_conf=CONF,
)
```

This will necessitate an additional section in the configuration file used.

```
[service_user]
auth_strategy = keystone
project_domain_name = Default
project_name = service
user_domain_name = Default
password = password
username = nova
auth_url = http://10.0.110.85/identity
auth_type = password
```

From existing CloudRegion

If you already have an *CloudRegion* you can pass it in instead:

```
from openstack import connection
import openstack.config

config = openstack.config.get_cloud_region(
    cloud='example',
    region_name='earth',
)
conn = connection.Connection(config=config)
```

Using the Connection

Services are accessed through an attribute named after the services official service-type.

List

An iterator containing a list of all the projects is retrieved in this manner:

```
projects = conn.identity.projects()
```

Find or create

If you wanted to make sure you had a network named zuul, you would first try to find it and if that fails, you would create it:

```
network = conn.network.find_network("zuul")
if network is None:
    network = conn.network.create_network(name="zuul")
```

Additional information about the services can be found in the *Service Proxies* documentation.

from_config

openstack.connection.**from_config**(*cloud=None*, *config=None*, *options=None*, ***kwargs*)

Create a Connection using openstack.config

Parameters

- **cloud** (*str*) Use the *cloud* configuration details when creating the Connection.
- **config** (openstack.config.cloud_region.CloudRegion) An existing CloudRegion configuration. If no *config* is provided, *openstack.config.OpenStackConfig* will be called, and the provided *name* will be used in determining which clouds configuration details will be used in creation of the *Connection* instance.
- **options** (*argparse.Namespace*) Allows direct passing in of options to be added to the cloud config. This does not have to be an actual instance of argparse.Namespace, despite the naming of the *openstack.config.loader.OpenStackConfig.get_one* argument to which it is passed.

Return type
Connection

Connection Object

class openstack.connection.**Connection**(*cloud=None*, *config=None*, *session=None*, *app_name=None*, *app_version=None*, *extra_services=None*, *strict=False*, *use_direct_get=None*, *task_manager=None*, *rate_limit=None*, *oslo_conf=None*, *service_types=None*, *global_request_id=None*, *strict_proxies=False*, *pool_executor=None*, ***kwargs*)

Create a connection to a cloud.

A connection needs information about how to connect, how to authenticate and how to select the appropriate services to use.

The recommended way to provide this information is by referencing a named cloud config from an existing *clouds.yaml* file. The cloud name `envvars` may be used to consume a cloud configured via `OS_` environment variables.

A pre-existing *CloudRegion* object can be passed in lieu of a cloud name, for cases where the user already has a fully formed CloudRegion and just wants to use it.

Similarly, if for some reason the user already has a `Session` and wants to use it, it may be passed in.

Parameters

- **cloud** (*str*) Name of the cloud from config to use.
- **config** (*CloudRegion*) CloudRegion object representing the config for the region of the cloud in question.
- **session** (`Session`) A session object compatible with `Session`.
- **app_name** (*str*) Name of the application to be added to User Agent.
- **app_version** (*str*) Version of the application to be added to User Agent.
- **extra_services** List of *ServiceDescription* objects describing services that openstacksdk otherwise does not know about.
- **use_direct_get** (*bool*) For get methods, make specific REST calls for server-side filtering instead of making list calls and filtering client-side. Default false.
- **task_manager** Ignored. Exists for backwards compat during transition. Rate limit parameters should be passed directly to the *rate_limit* parameter.
- **rate_limit** Client-side rate limit, expressed in calls per second. The parameter can either be a single float, or it can be a dict with keys as service-type and values as floats expressing the calls per second for that service. Defaults to None, which means no rate-limiting is performed.
- **oslo_conf** (`ConfigOpts`) An oslo.config `CONF` object that has been populated with `keystoneauth1.loading.register_adapter_conf_options` in groups named by the OpenStack services project name.

- **service_types** A list/set of service types this Connection should support. All other service types will be disabled (will error if used). **Currently only supported in conjunction with the "oslo_conf" kwarg.**
- **strict_proxies** (*bool*) Throw an `openstack.exceptions.ServiceDiscoveryException` if the endpoint for a given service doesnt work. This is useful for OpenStack services using sdk to talk to other OpenStack services where it can be expected that the deployer config is correct and errors should be reported immediately. Default false.
- **global_request_id** (*str*) A Request-id to send with all interactions.
- **pool_executor** (`Executor`) A futurist `Executor` object to be used for concurrent background activities. Defaults to None in which case a ThreadPoolExecutor will be created if needed.
- **kwargs** If a config is not provided, the rest of the parameters provided are assumed to be arguments to be passed to the CloudRegion constructor.

add_service(*service*)

Add a service to the Connection.

Attaches an instance of the *Proxy* class contained in *ServiceDescription*. The *Proxy* will be attached to the *Connection* by its `service_type` and by any `aliases` that may be specified.

Parameters

service (`openstack.service_description.ServiceDescription`) Object describing the service to be attached. As a convenience, if `service` is a string it will be treated as a `service_type` and a basic *ServiceDescription* will be created.

authorize()

Authorize this Connection

Note

This method is optional. When an application makes a call to any OpenStack service, this method allows you to request a token manually before attempting to do anything else.

Returns

A string token.

Raises

HttpException if the authorization fails due to reasons like the credentials provided are unable to be authorized or the *auth_type* argument is missing, etc.

connect_as(***kwargs*)

Make a new Connection object with new auth context.

Take the existing settings from the current cloud and construct a new Connection object with some of the auth settings overridden. This is useful for getting an object to perform tasks with as another user, or in the context of a different project.

```
conn = openstack.connect(cloud='example')
# Work normally
servers = conn.list_servers()
conn2 = conn.connect_as(username='different-user', password='')
# Work as different-user
servers = conn2.list_servers()
```

Parameters

kwargs keyword arguments can contain anything that would normally go in an auth dict. They will override the same settings from the parent cloud as appropriate. Entries that do not want to be overridden can be ommitted.

connect_as_project(*project*)

Make a new Connection object with a new project.

Take the existing settings from the current cloud and construct a new Connection object with the project settings overridden. This is useful for getting an object to perform tasks with as another user, or in the context of a different project.

```
cloud = openstack.connect(cloud='example')
# Work normally
servers = cloud.list_servers()
cloud2 = cloud.connect_as_project('different-project')
# Work in different-project
servers = cloud2.list_servers()
```

Parameters

project Either a project name or a project dict as returned by `list_projects`.

endpoint_for(*service_type*, *interface=None*, *region_name=None*)

Return the endpoint for a given service.

Respects config values for Connection, including `*_endpoint_override`. For direct values from the catalog regardless of overrides, see *get_endpoint_from_catalog()*

Parameters

- **service_type** Service Type of the endpoint to search for.
- **interface** Interface of the endpoint to search for. Optional, defaults to the configured value for interface for this Connection.
- **region_name** Region Name of the endpoint to search for. Optional, defaults to the configured value for region_name for this Connection.

Returns

The endpoint of the service, or None if not found.

add_auto_ip(*server*, *wait=False*, *timeout=60*, *reuse=True*)

Add a floating IP to a server.

This method is intended for basic usage. For advanced network architecture (e.g. multiple external networks or servers with multiple interfaces), use other floating IP methods.

This method can reuse available IPs, or allocate new IPs to the current project.

Parameters

- **server** a server dictionary.
- **reuse** Whether or not to attempt to reuse IPs, defaults to True.
- **wait** (optional) Wait for the address to appear as assigned to the server. Defaults to False.
- **timeout** (optional) Seconds to wait, defaults to 60. See the `wait` parameter.
- **reuse** Try to reuse existing ips. Defaults to True.

Returns

Floating IP address attached to server.

add_flavor_access(*flavor_id*, *project_id*)

Grant access to a private flavor for a project/tenant.

Parameters

- **flavor_id** (*string*) ID of the private flavor.
- **project_id** (*string*) ID of the project/tenant.

Returns

None

Raises

SDKException on operation error.

add_host_to_aggregate(*name_or_id*, *host_name*)

Add a host to an aggregate.

Parameters

- **name_or_id** Name or ID of the host aggregate.
- **host_name** Host to add.

Raises

SDKException on operation error.

add_ip_list(*server*, *ips*, *wait=False*, *timeout=60*, *fixed_address=None*, *nat_destination=None*)

Attach a list of IPs to a server.

Parameters

- **server** a server object
- **ips** list of floating IP addresses or a single address
- **wait** (optional) Wait for the address to appear as assigned to the server. Defaults to False.
- **timeout** (optional) Seconds to wait, defaults to 60. See the `wait` parameter.
- **fixed_address** (optional) Fixed address of the server to attach the IP to
- **nat_destination** (optional) Name or ID of the network that the fixed IP to attach the floating IP should be on

Returns
The updated server `openstack.compute.v2.server.Server`

Raises
SDKException on operation error.

add_router_interface(*router*, *subnet_id=None*, *port_id=None*)

Attach a subnet to an internal router interface.

Either a subnet ID or port ID must be specified for the internal interface. Supplying both will result in an error.

Parameters

- **router** (*dict*) The dict object of the router being changed
- **subnet_id** (*string*) The ID of the subnet to use for the interface
- **port_id** (*string*) The ID of the port to use for the interface

Returns
The raw response body from the request.

Raises
SDKException on operation error.

add_server_security_groups(*server*, *security_groups*)

Add security groups to a server.

Add existing security groups to an existing server. If the security groups are already present on the server this will continue unaffected.

Parameters

- **server** The server to remove security groups from.
- **security_groups** A list of security groups to remove.

Returns
False if server or security groups are undefined, True otherwise.

Raises
SDKException on operation error.

add_user_to_group(*name_or_id*, *group_name_or_id*)

Add a user to a group.

Parameters

- **name_or_id** Name or unique ID of the user.
- **group_name_or_id** Group name or ID

Raises
SDKException if something goes wrong during the OpenStack API call

add_volume_type_access(*name_or_id*, *project_id*)

Grant access on a volume_type to a project.

NOTE: the call works even if the project does not exist.

Parameters

- **name_or_id** ID or name of a volume_type
- **project_id** A project id

Returns
None

Raises
SDKException on operation error.

attach_port_to_machine(*name_or_id*, *port_name_or_id*)

Attach a virtual port to the bare metal machine.

Parameters

- **name_or_id** (*string*) A machine name or UUID.
- **port_name_or_id** (*string*) A port name or UUID. Note that this is a Network service port, not a bare metal NIC.

Returns
Nothing.

attach_volume(*server*, *volume*, *device=None*, *wait=True*, *timeout=None*)

Attach a volume to a server.

This will attach a volume, described by the passed in volume dict (as returned by get_volume()), to the server described by the passed in server dict (as returned by get_server()) on the named device on the server.

If the volume is already attached to the server, or generally not available, then an exception is raised. To re-attach to a server, but under a different device, the user must detach it first.

Parameters

- **server** The server dict to attach to.
- **volume** The volume dict to attach.
- **device** The device name where the volume will attach.
- **wait** If true, waits for volume to be attached.
- **timeout** Seconds to wait for volume attachment. None is forever.

Returns
a volume attachment object.

Raises
ResourceTimeout if wait time exceeded.

Raises
SDKException on operation error.

available_floating_ip(*network=None*, *server=None*)

Get a floating IP from a network or a pool.

Return the first available floating IP or allocate a new one.

Parameters

- **network** Name or ID of the network.

- **server** Server the IP is for if known

Returns
a (normalized) structure with a floating IP address description.

bind_accelerator_request(*uuid*, *properties*)

Bind an accelerator to VM.

Parameters

- **uuid** The uuid of the accelerator_request to be binded.
- **properties** The info of VM that will bind the accelerator.

Returns
True if bind succeeded, False otherwise.

close()

Release any resources held open.

create_accelerator_request(*attrs*)

Create an accelerator_request.

Parameters
attrs The info of accelerator_request to be created.

Returns
An accelerator `AcceleratorRequest` object.

create_aggregate(*name*, *availability_zone=None*)

Create a new host aggregate.

Parameters

- **name** Name of the host aggregate being created
- **availability_zone** Availability zone to assign hosts

Returns
The created compute `Aggregate` object.

Raises
SDKException on operation error.

create_cluster_template(*name*, *image_id=None*, *keypair_id=None*, *coe=None*, ***kwargs*)

Create a cluster template.

Parameters

- **name** (*string*) Name of the cluster template.
- **image_id** (*string*) Name or ID of the image to use.
- **keypair_id** (*string*) Name or ID of the keypair to use.
- **coe** (*string*) Name of the coe for the cluster template. Other arguments will be passed in kwargs.

Returns
a dict containing the cluster template description

Raises
SDKException if something goes wrong during the OpenStack API call

`create_coe_cluster`(*name*, *cluster_template_id*, ***kwargs*)

Create a COE cluster based on given cluster template.

Parameters

- **name** (*string*) Name of the cluster.
- **cluster_template_id** (*string*) ID of the cluster template to use.
- **kwargs** (*dict*) Any other arguments to pass in.

Returns
The created container infrastructure management `Cluster` object.

Raises
SDKException if something goes wrong during the OpenStack API call

`create_container`(*name*, *public=False*)

Create an object-store container.

Parameters

- **name** (*str*) Name of the container to create.
- **public** (*bool*) Whether to set this container to be public. Defaults to `False`.

Returns
The created object store `Container` object.

`create_device_profile`(*attrs*)

Create a device_profile.

Parameters
attrs The info of device_profile to be created.

Returns
An accelerator `DeviceProfile` objects.

`create_directory_marker_object`(*container*, *name*, ***headers*)

Create a zero-byte directory marker object

Note

This method is not needed in most cases. Modern swift does not require directory marker objects. However, some swift installs may need these.

When using swift Static Web and Web Listings to serve static content one may need to create a zero-byte object to represent each directory. Doing so allows Web Listings to generate an index of the objects inside of it, and allows Static Web to render index.html files that are inside the directory.

Parameters

- **container** The name of the container.

- **name** Name for the directory marker object within the container.
- **headers** These will be passed through to the object creation API as HTTP Headers.

Returns
The created object store `Object` object.

create_domain(*name*, *description=None*, *enabled=True*)

Create a domain.

Parameters

- **name** The name of the domain.
- **description** A description of the domain.
- **enabled** Is the domain enabled or not (default True).

Returns
The created identity `Endpoint` object.

Raises
SDKException if the domain cannot be created.

create_endpoint(*service_name_or_id*, *url=None*, *interface=None*, *region=None*, *enabled=True*, ***kwargs*)

Create a Keystone endpoint.

Parameters

- **service_name_or_id** Service name or id for this endpoint.
- **url** URL of the endpoint
- **interface** Interface type of the endpoint
- **public_url** Endpoint public URL.
- **internal_url** Endpoint internal URL.
- **admin_url** Endpoint admin URL.
- **region** Endpoint region.
- **enabled** Whether the endpoint is enabled

Returns
A list of identity `Endpoint` objects

Raises
SDKException if the service cannot be found or if something goes wrong during the OpenStack API call.

create_firewall_group(***kwargs*)

Creates firewall group. The keys egress_firewall_policy and ingress_firewall_policy are looked up and mapped as egress_firewall_policy_id and ingress_firewall_policy_id respectively. Port name or ids list is transformed to port ids list before the POST request.

Parameters

- **admin_state_up** (*bool*) State of firewall group. Will block all traffic if set to False. Defaults to True.

- **description** Human-readable description.
- **egress_firewall_policy** Name or id of egress firewall policy.
- **ingress_firewall_policy** Name or id of ingress firewall policy.
- **name** Human-readable name.
- **ports** (*list[str]*) List of associated ports (name or id)
- **project_id** Project id.
- **shared** Visibility to other projects. Defaults to False.

Raises
BadRequestException if parameters are malformed

Raises
DuplicateResource on multiple matches

Raises
NotFoundException if (ingress-, egress-) firewall policy or a port is not found.

Returns
The created network `FirewallGroup` object.

create_firewall_policy(***kwargs*)

Create firewall policy.

Parameters

- **audited** (*bool*) Status of audition of firewall policy. Set to False each time the firewall policy or the associated firewall rules are changed. Has to be explicitly set to True.
- **description** Human-readable description.
- **firewall_rules** (*list[str]*) List of associated firewall rules.
- **name** Human-readable name.
- **project_id** Project id.
- **shared** (*bool*) Visibility to other projects. Defaults to False.

Raises
BadRequestException if parameters are malformed

Raises
NotFoundException if a resource from firewall_list not found

Returns
The created network `FirewallPolicy` object.

create_firewall_rule(***kwargs*)

Creates firewall rule.

Parameters

- **action** Action performed on traffic. Valid values: allow, deny Defaults to deny.
- **description** Human-readable description.

- **destination_firewall_group_id** ID of destination firewall group.
- **destination_ip_address** IPv4-, IPv6 address or CIDR.
- **destination_port** Port or port range (e.g. 80:90)
- **enabled** (*bool*) Status of firewall rule. You can disable rules without disassociating them from firewall policies. Defaults to True.
- **ip_version** (*int*) IP Version. Valid values: 4, 6 Defaults to 4.
- **name** Human-readable name.
- **project_id** Project id.
- **protocol** IP protocol. Valid values: icmp, tcp, udp, null
- **shared** (*bool*) Visibility to other projects. Defaults to False.
- **source_firewall_group_id** ID of source firewall group.
- **source_ip_address** IPv4-, IPv6 address or CIDR.
- **source_port** Port or port range (e.g. 80:90)

Raises
BadRequestException if parameters are malformed

Returns
The created network `FirewallRule` object.

create_flavor(*name*, *ram*, *vcpus*, *disk*, *description=None*, *flavorid='auto'*, *ephemeral=0*, *swap=0*, *rxtx_factor=1.0*, *is_public=True*)

Create a new flavor.

Parameters

- **name** Descriptive name of the flavor
- **ram** Memory in MB for the flavor
- **vcpus** Number of VCPUs for the flavor
- **disk** Size of local disk in GB
- **description** Description of the flavor
- **flavorid** ID for the flavor (optional)
- **ephemeral** Ephemeral space size in GB
- **swap** Swap space in MB
- **rxtx_factor** RX/TX factor
- **is_public** Make flavor accessible to the public

Returns
The created compute `Flavor` object.

Raises
SDKException on operation error.

create_floating_ip(*network=None*, *server=None*, *fixed_address=None*, *nat_destination=None*, *port=None*, *wait=False*, *timeout=60*)

Allocate a new floating IP from a network or a pool.

Parameters

- **network** Name or ID of the network that the floating IP should come from.
- **server** (optional) Server dict for the server to create the IP for and to which it should be attached.
- **fixed_address** (optional) Fixed IP to attach the floating ip to.
- **nat_destination** (optional) Name or ID of the network that the fixed IP to attach the floating IP to should be on.
- **port** (optional) The port ID that the floating IP should be attached to. Specifying a port conflicts with specifying a server, fixed_address or nat_destination.
- **wait** (optional) Whether to wait for the IP to be active. Defaults to False. Only applies if a server is provided.
- **timeout** (optional) How long to wait for the IP to be active. Defaults to 60. Only applies if a server is provided.

Returns

a floating IP address

Raises

SDKException on operation error.

create_group(*name*, *description*, *domain=None*)

Create a group.

Parameters

- **name** (*string*) Group name.
- **description** (*string*) Group description.
- **domain** (*string*) Domain name or ID for the group.

Returns

An identity `Group` object

Raises

SDKException if something goes wrong during the OpenStack API call.

create_image(*name*, *filename=None*, *container=None*, *md5=None*, *sha256=None*, *disk_format=None*, *container_format=None*, *disable_vendor_agent=True*, *wait=False*, *timeout=3600*, *tags=None*, *allow_duplicates=False*, *meta=None*, *volume=None*, ***kwargs*)

Upload an image.

Parameters

- **name** (*str*) Name of the image to create. If it is a pathname of an image, the name will be constructed from the extensionless basename of the path.

- **filename** (*str*) The path to the file to upload, if needed. (optional, defaults to None)
- **container** (*str*) Name of the container in swift where images should be uploaded for import if the cloud requires such a thing. (optiona, defaults to images)
- **md5** (*str*) md5 sum of the image file. If not given, an md5 will be calculated.
- **sha256** (*str*) sha256 sum of the image file. If not given, an md5 will be calculated.
- **disk_format** (*str*) The disk format the image is in. (optional, defaults to the os-client-config config value for this cloud)
- **container_format** (*str*) The container format the image is in. (optional, defaults to the os-client-config config value for this cloud)
- **tags** (*list*) List of tags for this image. Each tag is a string of at most 255 chars.
- **disable_vendor_agent** (*bool*) Whether or not to append metadata flags to the image to inform the cloud in question to not expect a vendor agent to be runing. (optional, defaults to True)
- **wait** (*bool*) If true, waits for image to be created. Defaults to true - however, be aware that one of the upload methods is always synchronous.
- **timeout** Seconds to wait for image creation. None is forever.
- **allow_duplicates** If true, skips checks that enforce unique image name. (optional, defaults to False)
- **meta** A dict of key/value pairs to use for metadata that bypasses automatic type conversion.
- **volume** Name or ID or volume object of a volume to create an image from. Mutually exclusive with (optional, defaults to None)

Additional kwargs will be passed to the image creation as additional metadata for the image and will have all values converted to string except for min_disk, min_ram, size and virtual_size which will be converted to int.

If you are sure you have all of your data types correct or have an advanced need to be explicit, use meta. If you are just a normal consumer, using kwargs is likely the right choice.

If a value is in meta and kwargs, meta wins.

Returns
An image *openstack.image.v2.image.Image* object.

Raises
SDKException if there are problems uploading

create_image_snapshot(*name*, *server*, *wait=False*, *timeout=3600*, ***metadata*)

Create an image by snapshotting an existing server.

..note::
On most clouds this is a cold snapshot - meaning that the server in question will be shutdown before taking the snapshot. It is possible that its a live snapshot - but there is no way to know as a user, so caveat emptor.

Parameters

- **name** Name of the image to be created
- **server** Server name or ID or dict representing the server to be snapshotted
- **wait** If true, waits for image to be created.
- **timeout** Seconds to wait for image creation. None is forever.
- **metadata** Metadata to give newly-created image entity

Returns

The created image `Image` object.

Raises

SDKException if there are problems uploading

create_keypair(*name*, *public_key=None*)

Create a new keypair.

Parameters

- **name** Name of the keypair being created.
- **public_key** Public key for the new keypair.

Returns

The created compute `Keypair` object.

Raises

SDKException on operation error.

create_network(*name*, *shared=False*, *admin_state_up=True*, *external=False*, *provider=None*, *project_id=None*, *availability_zone_hints=None*, *port_security_enabled=None*, *mtu_size=None*, *dns_domain=None*)

Create a network.

Parameters

- **name** (*string*) Name of the network being created.
- **shared** (*bool*) Set the network as shared.
- **admin_state_up** (*bool*) Set the network administrative state to up.
- **external** (*bool*) Whether this network is externally accessible.
- **provider** (*dict*) A dict of network provider options. Example:

```
{'network_type': 'vlan', 'segmentation_id': 'vlan1'}
```

- **project_id** (*string*) Specify the project ID this network will be created on (admin-only).
- **availability_zone_hints** (*types.ListType*) A list of availability zone hints.
- **port_security_enabled** (*bool*) Enable / Disable port security
- **mtu_size** (*int*) maximum transmission unit value to address fragmentation. Minimum value is 68 for IPv4, and 1280 for IPv6.

- **dns_domain** (*string*) Specify the DNS domain associated with this network.

Returns
The created network `Network` object.

Raises
SDKException on operation error.

create_object(*container*, *name*, *filename=None*, *md5=None*, *sha256=None*, *segment_size=None*, *use_slo=True*, *metadata=None*, *generate_checksums=None*, *data=None*, ***headers*)

Create a file object.

Automatically uses large-object segments if needed.

Parameters

- **container** The name of the container to store the file in. This container will be created if it does not exist already.
- **name** Name for the object within the container.
- **filename** The path to the local file whose contents will be uploaded. Mutually exclusive with data.
- **data** The content to upload to the object. Mutually exclusive with filename.
- **md5** A hexadecimal md5 of the file. (Optional), if it is known and can be passed here, it will save repeating the expensive md5 process. It is assumed to be accurate.
- **sha256** A hexadecimal sha256 of the file. (Optional) See md5.
- **segment_size** Break the uploaded object into segments of this many bytes. (Optional) Shade will attempt to discover the maximum value for this from the server if it is not specified, or will use a reasonable default.
- **headers** These will be passed through to the object creation API as HTTP Headers.
- **use_slo** If the object is large enough to need to be a Large Object, use a static rather than dynamic object. Static Objects will delete segment objects when the manifest object is deleted. (optional, defaults to True)
- **generate_checksums** Whether to generate checksums on the client side that get added to headers for later prevention of double uploads of identical data. (optional, defaults to True)
- **metadata** This dict will get changed into headers that set metadata of the object

Returns
The created object store `Object` object.

Raises
SDKException on operation error.

create_port(*network_id*, ***kwargs*)

Create a port

Parameters

- **network_id** The ID of the network. (Required)
- **name** A symbolic name for the port. (Optional)
- **admin_state_up** The administrative status of the port, which is up (true, default) or down (false). (Optional)
- **mac_address** The MAC address. (Optional)
- **fixed_ips** List of ip_addresses and subnet_ids. See subnet_id and ip_address. (Optional) For example:

```
[
    {
        "ip_address": "10.29.29.13",
        "subnet_id": "a78484c4-c380-4b47-85aa-
↪21c51a2d8cbd",
    },
    ...,
]
```

- **subnet_id** If you specify only a subnet ID, OpenStack Networking allocates an available IP from that subnet to the port. (Optional) If you specify both a subnet ID and an IP address, OpenStack Networking tries to allocate the specified address to the port.
- **ip_address** If you specify both a subnet ID and an IP address, OpenStack Networking tries to allocate the specified address to the port.
- **security_groups** List of security group UUIDs. (Optional)
- **allowed_address_pairs** Allowed address pairs list (Optional) For example:

```
[
    {
        "ip_address": "23.23.23.1",
        "mac_address": "fa:16:3e:c4:cd:3f",
    },
    ...,
]
```

- **extra_dhcp_opts** Extra DHCP options. (Optional). For example:

```
[{"opt_name": "opt name1", "opt_value": "value1"}, ...]
```

- **device_owner** The ID of the entity that uses this port. For example, a DHCP agent. (Optional)
- **device_id** The ID of the device that uses this port. For example, a virtual server. (Optional)
- **vnic_type** (*binding*) The type of the created port. (Optional)
- **port_security_enabled** The security port state created on the network. (Optional)

- **qos_policy_id** The ID of the QoS policy to apply for port. (Optional)
- **project_id** The project in which to create the port. (Optional)
- **description** Description of the port. (Optional)
- **dns_domain** DNS domain relevant for the port. (Optional)
- **dns_name** DNS name of the port. (Optional)
- **numa_affinity_policy** the numa affinitiy policy. May be None, required, preferred or legacy. (Optional)
- **propagate_uplink_status** If the uplink status of the port should be propagated. (Optional)
- **mac_learning_enabled** If mac learning should be enabled on the port. (Optional)

Returns
The created network `Port` object.

Raises
SDKException on operation error.

create_project(*name*, *domain_id*, *description=None*, *enabled=True*, ***kwargs*)

Create a project.

Parameters

- **name**
- **domain_id**
- **description**
- **enabled**

Returns
An identity `Project` object.

create_qos_bandwidth_limit_rule(*policy_name_or_id*, *max_kbps*, ***kwargs*)

Create a QoS bandwidth limit rule.

Parameters

- **policy_name_or_id** (*string*) Name or ID of the QoS policy to which rule should be associated.
- **max_kbps** (*int*) Maximum bandwidth limit value (in kilobits per second).
- **max_burst_kbps** (*int*) Maximum burst value (in kilobits).
- **direction** (*string*) Ingress or egress. The direction in which the traffic will be limited.

Returns
The created network `QoSBandwidthLimitRule` object.

Raises
SDKException on operation error.

create_qos_dscp_marking_rule(*policy_name_or_id*, *dscp_mark*)

Create a QoS DSCP marking rule.

Parameters

- **policy_name_or_id** (*string*) Name or ID of the QoS policy to which rule should be associated.
- **dscp_mark** (*int*) DSCP mark value

Returns

The created network `QoSDSCPMarkingRule` object.

Raises

SDKException on operation error.

create_qos_minimum_bandwidth_rule(*policy_name_or_id*, *min_kbps*, ***kwargs*)

Create a QoS minimum bandwidth limit rule.

Parameters

- **policy_name_or_id** (*string*) Name or ID of the QoS policy to which rule should be associated.
- **min_kbps** (*int*) Minimum bandwidth value (in kilobits per second).
- **direction** (*string*) Ingress or egress. The direction in which the traffic will be available.

Returns

The created network `QoSMinimumBandwidthRule` object.

Raises

SDKException on operation error.

create_qos_policy(***kwargs*)

Create a QoS policy.

Parameters

- **name** (*string*) Name of the QoS policy being created.
- **description** (*string*) Description of created QoS policy.
- **shared** (*bool*) Set the QoS policy as shared.
- **default** (*bool*) Set the QoS policy as default for project.
- **project_id** (*string*) Specify the project ID this QoS policy will be created on (admin-only).

Returns

The created network `QosPolicy` object.

Raises

SDKException on operation error.

create_recordset(*zone*, *name*, *recordset_type*, *records*, *description=None*, *ttl=None*)

Create a recordset.

Parameters

- **zone** Name, ID or *openstack.dns.v2.zone.Zone* instance of the zone managing the recordset.
- **name** Name of the recordset
- **recordset_type** Type of the recordset
- **records** List of the recordset definitions
- **description** Description of the recordset
- **ttl** TTL value of the recordset

Returns
a dict representing the created recordset.

Raises
SDKException on operation error.

create_role(*name*, ***kwargs*)

Create a Keystone role.

Parameters

- **name** (*string*) The name of the role.
- **domain_id** domain id (v3)

Returns
an identity `Role` object

Raises
SDKException if the role cannot be created

create_router(*name=None*, *admin_state_up=True*, *ext_gateway_net_id=None*, *enable_snat=None*, *ext_fixed_ips=None*, *project_id=None*, *availability_zone_hints=None*)

Create a logical router.

Parameters

- **name** (*string*) The router name.
- **admin_state_up** (*bool*) The administrative state of the router.
- **ext_gateway_net_id** (*string*) Network ID for the external gateway.
- **enable_snat** (*bool*) Enable Source NAT (SNAT) attribute.
- **ext_fixed_ips** List of dictionaries of desired IP and/or subnet on the external network. Example:

```
[
    {
        "subnet_id": "8ca37218-28ff-41cb-9b10-
↪039601ea7e6b",
        "ip_address": "192.168.10.2",
    }
]
```

- **project_id** (*string*) Project ID for the router.

- **availability_zone_hints** (*types.ListType*) A list of availability zone hints.

Returns
The created network `Router` object.

Raises
SDKException on operation error.

create_security_group(*name*, *description*, *project_id=None*, *stateful=None*)

Create a new security group

Parameters

- **name** (*string*) A name for the security group.
- **description** (*string*) Describes the security group.
- **project_id** (*string*) Specify the project ID this security group will be created on (admin-only).
- **stateful** (*string*) Whether the security group is stateful or not.

Returns
A `openstack.network.v2.security_group.SecurityGroup` representing the new security group.

Raises
SDKException on operation error.

Raises
OpenStackCloudUnavailableFeature if security groups are not supported on this cloud.

create_security_group_rule(*secgroup_name_or_id*, *port_range_min=None*, *port_range_max=None*, *protocol=None*, *remote_ip_prefix=None*, *remote_group_id=None*, *remote_address_group_id=None*, *direction='ingress'*, *ethertype='IPv4'*, *project_id=None*, *description=None*)

Create a new security group rule

Parameters

- **secgroup_name_or_id** (*string*) The security group name or ID to associate with this security group rule. If a non-unique group name is given, an exception is raised.
- **port_range_min** (*int*) The minimum port number in the range that is matched by the security group rule. If the protocol is TCP or UDP, this value must be less than or equal to the port_range_max attribute value. If nova is used by the cloud provider for security groups, then a value of None will be transformed to -1.
- **port_range_max** (*int*) The maximum port number in the range that is matched by the security group rule. The port_range_min attribute constrains the port_range_max attribute. If nova is used by the cloud provider for security groups, then a value of None will be transformed to -1.

- **protocol** (*string*) The protocol that is matched by the security group rule. Valid values are None, tcp, udp, and icmp.
- **remote_ip_prefix** (*string*) The remote IP prefix to be associated with this security group rule. This attribute matches the specified IP prefix as the source IP address of the IP packet.
- **remote_group_id** (*string*) The remote group ID to be associated with this security group rule.
- **remote_address_group_id** (*string*) The remote address group ID to be associated with this security group rule.
- **direction** (*string*) Ingress or egress: The direction in which the security group rule is applied. For a compute instance, an ingress security group rule is applied to incoming (ingress) traffic for that instance. An egress rule is applied to traffic leaving the instance.
- **ethertype** (*string*) Must be IPv4 or IPv6, and addresses represented in CIDR must match the ingress or egress rules.
- **project_id** (*string*) Specify the project ID this security group will be created on (admin-only).
- **description** (*string*) Description of the rule, max 255 characters.

Returns

A `openstack.network.v2.security_group.SecurityGroup` representing the new security group rule.

Raises

SDKException on operation error.

create_server(*name*, *image=None*, *flavor=None*, *auto_ip=True*, *ips=None*, *ip_pool=None*, *root_volume=None*, *terminate_volume=False*, *wait=False*, *timeout=180*, *reuse_ips=True*, *network=None*, *boot_from_volume=False*, *volume_size='50'*, *boot_volume=None*, *volumes=None*, *nat_destination=None*, *group=None*, ***kwargs*)

Create a virtual server instance.

Parameters

- **name** Something to name the server.
- **image** Image dict, name or ID to boot with. image is required unless boot_volume is given.
- **flavor** Flavor dict, name or ID to boot onto.
- **auto_ip** Whether to take actions to find a routable IP for the server. (defaults to True)
- **ips** List of IPs to attach to the server (defaults to None)
- **ip_pool** Name of the network or floating IP pool to get an address from. (defaults to None)
- **root_volume** Name or ID of a volume to boot from (defaults to None - deprecated, use boot_volume)

- **boot_volume** Name or ID of a volume to boot from (defaults to None)
- **terminate_volume** If booting from a volume, whether it should be deleted when the server is destroyed. (defaults to False)
- **volumes** (optional) A list of volumes to attach to the server
- **meta** (optional) A dict of arbitrary key/value metadata to store for this server. Both keys and values must be <=255 characters.
- **files** (optional, deprecated) A dict of files to overwrite on the server upon boot. Keys are file names (i.e. `/etc/passwd`) and values are the file contents (either as a string or as a file-like object). A maximum of five entries is allowed, and each file must be 10k or less.
- **reservation_id** a UUID for the set of servers being requested.
- **min_count** (optional extension) The minimum number of servers to launch.
- **max_count** (optional extension) The maximum number of servers to launch.
- **security_groups** A list of security group names
- **userdata** user data to pass to be exposed by the metadata server this can be a file type object as well or a string.
- **key_name** (optional extension) name of previously created keypair to inject into the instance.
- **availability_zone** Name of the availability zone for instance placement.
- **block_device_mapping** (optional) A dict of block device mappings for this server.
- **block_device_mapping_v2** (optional) A dict of block device mappings for this server.
- **nics** (optional extension) an ordered list of nics to be added to this server, with information about connected networks, fixed IPs, port etc.
- **scheduler_hints** (optional extension) arbitrary key-value pairs specified by the client to help boot an instance
- **config_drive** (optional extension) value for config drive either boolean, or volume-id
- **disk_config** (optional extension) control how the disk is partitioned when the server is created. possible values are AUTO or MANUAL.
- **admin_pass** (optional extension) add a user supplied admin password.
- **wait** (optional) Wait for the address to appear as assigned to the server. Defaults to False.
- **timeout** (optional) Seconds to wait, defaults to 60. See the `wait` parameter.
- **reuse_ips** (optional) Whether to attempt to reuse pre-existing floating ips should a floating IP be needed (defaults to True)
- **network** (optional) Network dict or name or ID to attach the server to. Mutually exclusive with the nics parameter. Can also be a list of network names or IDs or network dicts.

- **boot_from_volume** Whether to boot from volume. boot_volume implies True, but boot_from_volume=True with no boot_volume is valid and will create a volume from the image and use that.
- **volume_size** When booting an image from volume, how big should the created volume be? Defaults to 50.
- **nat_destination** Which network should a created floating IP be attached to, if its not possible to infer from the clouds configuration. (Optional, defaults to None)
- **group** ServerGroup dict, name or id to boot the server in. If a group is provided in both scheduler_hints and in the group param, the group param will win. (Optional, defaults to None)

Returns
The created compute `Server` object.

Raises
SDKException on operation error.

create_server_group(*name*, *policies=None*, *policy=None*)

Create a new server group.

Parameters

- **name** Name of the server group being created
- **policies** List of policies for the server group.

Returns
The created compute `ServerGroup` object.

Raises
SDKException on operation error.

create_service(*name*, *enabled=True*, ***kwargs*)

Create a service.

Parameters

- **name** Service name.
- **type** Service type. (type or service_type required.)
- **service_type** Service type. (type or service_type required.)
- **description** Service description (optional).
- **enabled** Whether the service is enabled (v3 only)

Returns
an identity `Service` object

Raises
SDKException if something goes wrong during the OpenStack API call.

create_stack(*name*, *tags=None*, *template_file=None*, *template_url=None*, *template_object=None*, *files=None*, *rollback=True*, *wait=False*, *timeout=3600*, *environment_files=None*, ***parameters*)

Create a stack.

Parameters

- **name** (*string*) Name of the stack.
- **tags** List of tag(s) of the stack. (optional)
- **template_file** (*string*) Path to the template.
- **template_url** (*string*) URL of template.
- **template_object** (*string*) URL to retrieve template object.
- **files** (*dict*) dict of additional file content to include.
- **rollback** (*boolean*) Enable rollback on create failure.
- **wait** (*boolean*) Whether to wait for the delete to finish.
- **timeout** (*int*) Stack create timeout in seconds.
- **environment_files** Paths to environment files to apply.

Other arguments will be passed as stack parameters which will take precedence over any parameters specified in the environments.

Only one of template_file, template_url, template_object should be specified.

Returns
a dict containing the stack description

Raises
SDKException if something goes wrong during the OpenStack API call

create_subnet(*network_name_or_id*, *cidr=None*, *ip_version=4*, *enable_dhcp=False*, *subnet_name=None*, *tenant_id=None*, *allocation_pools=None*, *gateway_ip=None*, *disable_gateway_ip=False*, *dns_nameservers=None*, *host_routes=None*, *ipv6_ra_mode=None*, *ipv6_address_mode=None*, *prefixlen=None*, *use_default_subnetpool=False*, *subnetpool_name_or_id=None*, ***kwargs*)

Create a subnet on a specified network.

Parameters

- **network_name_or_id** (*string*) The unique name or ID of the attached network. If a non-unique name is supplied, an exception is raised.
- **cidr** (*string*) The CIDR. Only one of `cidr`, `use_default_subnetpool` and `subnetpool_name_or_id` may be specified at the same time.
- **ip_version** (*int*) The IP version, which is 4 or 6.
- **enable_dhcp** (*bool*) Set to `True` if DHCP is enabled and `False` if disabled. Default is `False`.
- **subnet_name** (*string*) The name of the subnet.
- **tenant_id** (*string*) The ID of the tenant who owns the network. Only administrative users can specify a tenant ID other than their own.
- **allocation_pools** A list of dictionaries of the start and end addresses for the allocation pools. For example:

```
[{"start": "192.168.199.2", "end": "192.168.199.254"}]
```

- **gateway_ip** (*string*) The gateway IP address. When you specify both allocation_pools and gateway_ip, you must ensure that the gateway IP does not overlap with the specified allocation pools.
- **disable_gateway_ip** (*bool*) Set to `True` if gateway IP address is disabled and `False` if enabled. It is not allowed with gateway_ip. Default is `False`.
- **dns_nameservers** A list of DNS name servers for the subnet. For example:

```
["8.8.8.7", "8.8.8.8"]
```

- **host_routes** A list of host route dictionaries for the subnet. For example:

```
[
    {"destination": "0.0.0.0/0", "nexthop": "123.456.78.9
↪"},
    {"destination": "192.168.0.0/24", "nexthop": "192.
↪168.0.1"},
]
```

- **ipv6_ra_mode** (*string*) IPv6 Router Advertisement mode. Valid values are: dhcpv6-stateful, dhcpv6-stateless, or slaac.
- **ipv6_address_mode** (*string*) IPv6 address mode. Valid values are: dhcpv6-stateful, dhcpv6-stateless, or slaac.
- **prefixlen** (*string*) The prefix length to use for subnet allocation from a subnetpool.
- **use_default_subnetpool** (*bool*) Use the default subnetpool for `ip_version` to obtain a CIDR. Only one of `cidr`, `use_default_subnetpool` and `subnetpool_name_or_id` may be specified at the same time.
- **subnetpool_name_or_id** (*string*) The unique name or id of the subnetpool to obtain a CIDR from. Only one of `cidr`, `use_default_subnetpool` and `subnetpool_name_or_id` may be specified at the same time.
- **kwargs** Key value pairs to be passed to the Neutron API.

Returns

The created network `Subnet` object.

Raises

SDKException on operation error.

create_user(*name*, *password=None*, *email=None*, *default_project=None*, *enabled=True*, *domain_id=None*, *description=None*)

Create a user.

create_volume(*size*, *wait=True*, *timeout=None*, *image=None*, *bootable=None*, ***kwargs*)

Create a volume.

Parameters

- **size** Size, in GB of the volume to create.
- **wait** If true, waits for volume to be created.
- **timeout** Seconds to wait for volume creation. None is forever.
- **image** (optional) Image name, ID or object from which to create the volume
- **bootable** (optional) Make this volume bootable. If set, wait will also be set to true.
- **kwargs** Keyword arguments as expected for cinder client.

Returns
The created volume `Volume` object.

Raises
ResourceTimeout if wait time exceeded.

Raises
SDKException on operation error.

create_volume_backup(*volume_id*, *name=None*, *description=None*, *force=False*, *wait=True*, *timeout=None*, *incremental=False*, *snapshot_id=None*)

Create a volume backup.

Parameters

- **volume_id** the ID of the volume to backup.
- **name** name of the backup, one will be generated if one is not provided
- **description** description of the backup, one will be generated if one is not provided
- **force** If set to True the backup will be created even if the volume is attached to an instance, if False it will not
- **wait** If true, waits for volume backup to be created.
- **timeout** Seconds to wait for volume backup creation. None is forever.
- **incremental** If set to true, the backup will be incremental.
- **snapshot_id** The UUID of the source snapshot to back up.

Returns
The created volume `Backup` object.

Raises
ResourceTimeout if wait time exceeded.

Raises
SDKException on operation error.

create_volume_snapshot(*volume_id*, *force=False*, *wait=True*, *timeout=None*, ***kwargs*)

Create a volume.

Parameters

- **volume_id** the ID of the volume to snapshot.

- **force** If set to True the snapshot will be created even if the volume is attached to an instance, if False it will not
- **name** name of the snapshot, one will be generated if one is not provided
- **description** description of the snapshot, one will be generated if one is not provided
- **wait** If true, waits for volume snapshot to be created.
- **timeout** Seconds to wait for volume snapshot creation. None is forever.

Returns
The created volume `Snapshot` object.

Raises
ResourceTimeout if wait time exceeded.

Raises
SDKException on operation error.

create_zone(*name*, *zone_type=None*, *email=None*, *description=None*, *ttl=None*, *masters=None*)

Create a new zone.

Parameters

- **name** Name of the zone being created.
- **zone_type** Type of the zone (primary/secondary)
- **email** Email of the zone owner (only applies if zone_type is primary)
- **description** Description of the zone
- **ttl** TTL (Time to live) value in seconds
- **masters** Master nameservers (only applies if zone_type is secondary)

Returns
a dict representing the created zone.

Raises
SDKException on operation error.

property current_location

Return a `utils.Munch` explaining the current cloud location.

property current_project

Return a `utils.Munch` describing the current project

property current_project_id

Get the current project ID.

Returns the project_id of the current token scope. None means that the token is domain scoped or unscoped.

Raises

- **keystoneauth1.exceptions.auth.AuthorizationFailure** if a new token fetch fails.

- **keystoneauth1.exceptions.auth_plugins.MissingAuthPlugin** if a plugin is not available.

property current_user_id

Get the id of the currently logged-in user from the token.

delete_accelerator_request(*name_or_id*, *filters*)

Delete a accelerator_request.

Parameters

- **name_or_id** The name or UUID of the accelerator request to be deleted.
- **filters** dict of filter conditions to push down

Returns

True if delete succeeded, False otherwise.

delete_aggregate(*name_or_id*)

Delete a host aggregate.

Parameters

name_or_id Name or ID of the host aggregate to delete.

Returns

True if delete succeeded, False otherwise.

Raises

SDKException on operation error.

delete_autocreated_image_objects(*container=None*, *segment_prefix=None*)

Delete all objects autocreated for image uploads.

This method should generally not be needed, as shade should clean up the objects it uses for object-based image creation. If something goes wrong and it is found that there are leaked objects, this method can be used to delete any objects that shade has created on the users behalf in service of image uploads.

Parameters

- **container** (*str*) Name of the container. Defaults to images.
- **segment_prefix** (*str*) Prefix for the image segment names to delete. If not given, all image upload segments present are deleted.

Returns

True if deletion was successful, else False.

delete_cluster_template(*name_or_id*)

Delete a cluster template.

Parameters

name_or_id Name or unique ID of the cluster template.

Returns

True if the delete succeeded, False if the cluster template was not found.

Raises

SDKException on operation error.

delete_coe_cluster(*name_or_id*)

Delete a COE cluster.

Parameters
name_or_id Name or unique ID of the cluster.

Returns
True if the delete succeeded, False if the cluster was not found.

Raises
SDKException on operation error.

delete_compute_quotas(*name_or_id*)

Delete quota for a project

Parameters
name_or_id project name or id

Raises
SDKException if its not a valid project or the nova client call failed

delete_container(*name*)

Delete an object-store container.

Parameters
name (*str*) Name of the container to delete.

delete_device_profile(*name_or_id*, *filters*)

Delete a device_profile.

Parameters

- **name_or_id** The name or uuid of the device profile to be deleted.
- **filters** dict of filter conditions to push down

Returns
True if delete succeeded, False otherwise.

delete_domain(*domain_id=None*, *name_or_id=None*)

Delete a Keystone domain.

Parameters

- **domain_id** ID of the domain to delete.
- **name_or_id** Name or unique ID of the domain.

Returns
True if delete succeeded, False otherwise.

Raises
SDKException if something goes wrong during the OpenStack API call.

delete_endpoint(*id*)

Delete a Keystone endpoint.

Parameters
id ID of the endpoint to delete.

Returns
True if delete succeeded, False otherwise.

Raises
SDKException if something goes wrong during the OpenStack API call.

delete_firewall_group(*name_or_id*, *filters=None*)

Deletes firewall group. Prints debug message in case to-be-deleted resource was not found.

Parameters

- **name_or_id** firewall group name or id
- **filters** (*dict*) optional filters

Raises
DuplicateResource on multiple matches

Returns
True if resource is successfully deleted, False otherwise.

Return type
bool

delete_firewall_policy(*name_or_id*, *filters=None*)

Deletes firewall policy. Prints debug message in case to-be-deleted resource was not found.

Parameters

- **name_or_id** firewall policy name or id
- **filters** A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR A string containing a jmespath expression for further filtering. Example:: [?last_name==‘Smith‘] | [?other.gender]==‘Female‘]

Raises
DuplicateResource on multiple matches

Returns
True if resource is successfully deleted, False otherwise.

Return type
bool

delete_firewall_rule(*name_or_id*, *filters=None*)

Deletes firewall rule.

Prints debug message in case to-be-deleted resource was not found.

Parameters

- **name_or_id** firewall rule name or id
- **filters** A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR A string containing a jmespath expression for further filtering. Example:: [?last_name==‘Smith‘] | [?other.gender]==‘Female‘]

Raises
DuplicateResource on multiple matches

Returns
True if resource is successfully deleted, False otherwise.

Return type
bool

delete_flavor(*name_or_id*)

Delete a flavor

Parameters
name_or_id ID or name of the flavor to delete.

Returns
True if delete succeeded, False otherwise.

Raises
SDKException on operation error.

delete_floating_ip(*floating_ip_id*, *retry=1*)

Deallocate a floating IP from a project.

Parameters

- **floating_ip_id** a floating IP address ID.
- **retry** number of times to retry. Optional, defaults to 1, which is in addition to the initial delete call. A value of 0 will also cause no checking of results to occur.

Returns
True if the IP address has been deleted, False if the IP address was not found.

Raises
SDKException on operation error.

delete_group(*name_or_id*)

Delete a group

Parameters
name_or_id Name or unique ID of the group.

Returns
True if delete succeeded, False otherwise.

Raises
SDKException if something goes wrong during the OpenStack API call.

delete_image(*name_or_id*, *wait=False*, *timeout=3600*, *delete_objects=True*)

Delete an existing image.

Parameters

- **name_or_id** Name of the image to be deleted.
- **wait** If True, waits for image to be deleted.
- **timeout** Seconds to wait for image deletion. None is forever.
- **delete_objects** If True, also deletes uploaded swift objects.

Returns
True if delete succeeded, False otherwise.

Raises
SDKException if there are problems deleting.

delete_keypair(*name*)

Delete a keypair.

Parameters
name Name of the keypair to delete.

Returns
True if delete succeeded, False otherwise.

Raises
SDKException on operation error.

delete_network(*name_or_id*)

Delete a network.

Parameters
name_or_id Name or ID of the network being deleted.

Returns
True if delete succeeded, False otherwise.

Raises
SDKException on operation error.

delete_network_quotas(*name_or_id*)

Delete network quotas for a project

Parameters
name_or_id project name or id

Returns
dict with the quotas

Raises
SDKException if its not a valid project or the network client call failed

delete_object(*container*, *name*, *meta=None*)

Delete an object from a container.

Parameters

- **container** (*string*) Name of the container holding the object.
- **name** (*string*) Name of the object to delete.
- **meta** (*dict*) Metadata for the object in question. (optional, will be fetched if not provided)

Returns
True if delete succeeded, False if the object was not found.

Raises
SDKException on operation error.

delete_port(*name_or_id*)

Delete a port

Parameters
name_or_id ID or name of the port to delete.

Returns
True if delete succeeded, False otherwise.

Raises
SDKException on operation error.

delete_project(*name_or_id*, *domain_id=None*)

Delete a project.

Parameters

- **name_or_id** Name or unique ID of the project.
- **domain_id** Domain ID to scope the retrieved project.

Returns
True if delete succeeded, False if the project was not found.

Raises
SDKException if something goes wrong during the OpenStack API call

delete_qos_bandwidth_limit_rule(*policy_name_or_id*, *rule_id*)

Delete a QoS bandwidth limit rule.

Parameters

- **policy_name_or_id** (*string*) Name or ID of the QoS policy to which rule is associated.
- **rule_id** (*string*) ID of rule to update.

Raises
SDKException on operation error.

delete_qos_dscp_marking_rule(*policy_name_or_id*, *rule_id*)

Delete a QoS DSCP marking rule.

Parameters

- **policy_name_or_id** (*string*) Name or ID of the QoS policy to which rule is associated.
- **rule_id** (*string*) ID of rule to update.

Raises
SDKException on operation error.

delete_qos_minimum_bandwidth_rule(*policy_name_or_id*, *rule_id*)

Delete a QoS minimum bandwidth rule.

Parameters

- **policy_name_or_id** (*string*) Name or ID of the QoS policy to which rule is associated.
- **rule_id** (*string*) ID of rule to delete.

Raises

SDKException on operation error.

delete_qos_policy(*name_or_id*)

Delete a QoS policy.

Parameters

name_or_id Name or ID of the policy being deleted.

Returns

True if delete succeeded, False otherwise.

Raises

SDKException on operation error.

delete_recordset(*zone*, *name_or_id*)

Delete a recordset.

Parameters

- **zone** Name, ID or *openstack.dns.v2.zone.Zone* instance of the zone managing the recordset.
- **name_or_id** Name or ID of the recordset being deleted.

Returns

True if delete succeeded, False otherwise.

Raises

SDKException on operation error.

delete_role(*name_or_id*, ***kwargs*)

Delete a Keystone role.

Parameters

- **name_or_id** Name or unique ID of the role.
- **domain_id** domain id (v3)

Returns

True if delete succeeded, False otherwise.

Raises

SDKException if something goes wrong during the OpenStack API call.

delete_router(*name_or_id*)

Delete a logical router.

If a name, instead of a unique UUID, is supplied, it is possible that we could find more than one matching router since names are not required to be unique. An error will be raised in this case.

Parameters
name_or_id Name or ID of the router being deleted.

Returns
True if delete succeeded, False otherwise.

Raises
SDKException on operation error.

delete_security_group(*name_or_id*)

Delete a security group

Parameters
name_or_id (*string*) The name or unique ID of the security group.

Returns
True if delete succeeded, False otherwise.

Raises
SDKException on operation error.

Raises
OpenStackCloudUnavailableFeature if security groups are not supported on this cloud.

delete_security_group_rule(*rule_id*)

Delete a security group rule

Parameters
rule_id (*string*) The unique ID of the security group rule.

Returns
True if delete succeeded, False otherwise.

Raises
SDKException on operation error.

Raises
OpenStackCloudUnavailableFeature if security groups are not supported on this cloud.

delete_server(*name_or_id*, *wait=False*, *timeout=180*, *delete_ips=False*, *delete_ip_retry=1*)

Delete a server instance.

Parameters

- **name_or_id** name or ID of the server to delete
- **wait** (*bool*) If true, waits for server to be deleted.
- **timeout** (*int*) Seconds to wait for server deletion.
- **delete_ips** (*bool*) If true, deletes any floating IPs associated with the instance.
- **delete_ip_retry** (*int*) Number of times to retry deleting any floating ips, should the first try be unsuccessful.

Returns
True if delete succeeded, False otherwise if the server does not exist.

Raises
SDKException on operation error.

delete_server_group(*name_or_id*)

Delete a server group.

Parameters
name_or_id Name or ID of the server group to delete

Returns
True if delete succeeded, False otherwise

Raises
SDKException on operation error.

delete_server_metadata(*name_or_id*, *metadata_keys*)

Delete metadata from a server instance.

Parameters

- **name_or_id** (*str*) The name or ID of the server instance to update.
- **metadata_keys** A list with the keys to be deleted from the server instance.

Returns
None

Raises
SDKException on operation error.

delete_service(*name_or_id*)

Delete a Keystone service.

Parameters
name_or_id Name or unique ID of the service.

Returns
True if delete succeeded, False otherwise.

Raises
SDKException if something goes wrong during the OpenStack API call

delete_stack(*name_or_id*, *wait=False*)

Delete a stack

Parameters

- **name_or_id** (*string*) Stack name or ID.
- **wait** (*boolean*) Whether to wait for the delete to finish

Returns
True if delete succeeded, False if the stack was not found.

Raises
SDKException if something goes wrong during the OpenStack API call

delete_subnet(*name_or_id*)

Delete a subnet.

If a name, instead of a unique UUID, is supplied, it is possible that we could find more than one matching subnet since names are not required to be unique. An error will be raised in this case.

Parameters
name_or_id Name or ID of the subnet being deleted.

Returns
True if delete succeeded, False otherwise.

Raises
SDKException on operation error.

delete_unattached_floating_ips(*retry=1*)

Safely delete unattached floating ips.

If the cloud can safely purge any unattached floating ips without race conditions, do so.

Safely here means a specific thing. It means that you are not running this while another process that might do a two step create/attach is running. You can safely run this method while another process is creating servers and attaching floating IPs to them if either that process is using add_auto_ip from shade, or is creating the floating IPs by passing in a server to the create_floating_ip call.

Parameters
retry number of times to retry. Optional, defaults to 1, which is in addition to the initial delete call. A value of 0 will also cause no checking of results to occur.

Returns
Number of Floating IPs deleted, False if none

Raises
SDKException on operation error.

delete_volume(*name_or_id=None*, *wait=True*, *timeout=None*, *force=False*)

Delete a volume.

Parameters

- **name_or_id** Name or unique ID of the volume.
- **wait** If true, waits for volume to be deleted.
- **timeout** Seconds to wait for volume deletion. None is forever.
- **force** Force delete volume even if the volume is in deleting or error_deleting state.

Returns
True if deletion was successful, else False.

Raises
ResourceTimeout if wait time exceeded.

Raises
SDKException on operation error.

delete_volume_backup(*name_or_id=None*, *force=False*, *wait=False*, *timeout=None*)

Delete a volume backup.

Parameters

- **name_or_id** Name or unique ID of the volume backup.
- **force** Allow delete in state other than error or available.
- **wait** If true, waits for volume backup to be deleted.
- **timeout** Seconds to wait for volume backup deletion. None is forever.

Returns

True if deletion was successful, else False.

Raises

ResourceTimeout if wait time exceeded.

Raises

SDKException on operation error.

delete_volume_quotas(*name_or_id*)

Delete volume quotas for a project

Parameters

name_or_id project name or id

Returns

The deleted volume `QuotaSet` object.

Raises

SDKException if its not a valid project or the call failed

delete_volume_snapshot(*name_or_id=None*, *wait=False*, *timeout=None*)

Delete a volume snapshot.

Parameters

- **name_or_id** Name or unique ID of the volume snapshot.
- **wait** If true, waits for volume snapshot to be deleted.
- **timeout** Seconds to wait for volume snapshot deletion. None is forever.

Returns

True if deletion was successful, else False.

Raises

ResourceTimeout if wait time exceeded.

Raises

SDKException on operation error.

delete_zone(*name_or_id*)

Delete a zone.

Parameters

name_or_id Name or ID of the zone being deleted.

Returns

True if delete succeeded, False otherwise.

Raises
SDKException on operation error.

detach_ip_from_server(*server_id*, *floating_ip_id*)

Detach a floating IP from a server.

Parameters

- **server_id** ID of a server.
- **floating_ip_id** Id of the floating IP to detach.

Returns
True if the IP has been detached, or False if the IP wasnt attached to any server.

Raises
SDKException on operation error.

detach_port_from_machine(*name_or_id*, *port_name_or_id*)

Detach a virtual port from the bare metal machine.

Parameters

- **name_or_id** (*string*) A machine name or UUID.
- **port_name_or_id** (*string*) A port name or UUID. Note that this is a Network service port, not a bare metal NIC.

Returns
Nothing.

detach_volume(*server*, *volume*, *wait=True*, *timeout=None*)

Detach a volume from a server.

Parameters

- **server** The server dict to detach from.
- **volume** The volume dict to detach.
- **wait** If true, waits for volume to be detached.
- **timeout** Seconds to wait for volume detachment. None is forever.

Returns
None

Raises
ResourceTimeout if wait time exceeded.

Raises
SDKException on operation error.

download_image(*name_or_id*, *output_path=None*, *output_file=None*, *chunk_size=1048576*)

Download an image by name or ID

Parameters

- **name_or_id** (*str*) Name or ID of the image.
- **output_path** the output path to write the image to. Either this or output_file must be specified

- **output_file** a file object (or file-like object) to write the image data to. Only write() will be called on this object. Either this or output_path must be specified
- **chunk_size** (*int*) size in bytes to read from the wire and buffer at one time. Defaults to 1024 * 1024 = 1 MiB

Returns
When output_path and output_file are not given - the bytes comprising the given Image when stream is False, otherwise a `requests.Response` instance. When output_path or output_file are given - an image *Image* instance.

Raises
SDKException in the event download_image is called without exactly one of either output_path or output_file

Raises
BadRequestException if no images are found matching the name or ID provided

get_aggregate(*name_or_id*, *filters=None*)

Get an aggregate by name or ID.

Parameters

- **name_or_id** Name or ID of the aggregate.
- **filters** (*dict*) **DEPRECATED** A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{
    'availability_zone': 'nova',
    'metadata': {'cpu_allocation_ratio': '1.0'},
}
```

Returns
An aggregate dict or None if no matching aggregate is found.

get_cluster_template(*name_or_id*, *filters=None*, *detail=False*)

Get a cluster template by name or ID.

Parameters

- **name_or_id** Name or ID of the cluster template.
- **filters** A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR A string containing a jmespath expression for further filtering. Example:: [?last_name==‘Smith‘] | [?other.gender]==‘Female‘]

Returns
A cluster template dict or None if no matching cluster template is found.

get_coe_cluster(*name_or_id*, *filters=None*)

Get a COE cluster by name or ID.

Parameters

- **name_or_id** Name or ID of the cluster.
- **filters** A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR A string containing a jmespath expression for further filtering. Example:: [?last_name==‘Smith‘] | [?other.gender]==‘Female‘]

Returns

A container infrastructure management `Cluster` object if found, else None.

get_coe_cluster_certificate(*cluster_id*)

Get details about the CA certificate for a cluster by name or ID.

Parameters

cluster_id ID of the cluster.

Returns

Details about the CA certificate for the given cluster.

get_compute_limits(*name_or_id=None*)

Get absolute compute limits for a project

Parameters

name_or_id (optional) project name or ID to get limits for if different from the current project

Returns

A compute `AbsoluteLimits` object.

Raises

SDKException if its not a valid project

get_compute_quotas(*name_or_id*)

Get quota for a project

Parameters

name_or_id project name or id

Returns

A compute `QuotaSet` object if found, else None.

Raises

SDKException if its not a valid project

get_compute_usage(*name_or_id*, *start=None*, *end=None*)

Get usage for a specific project

Parameters

- **name_or_id** project name or id

- **start** `datetime.datetime` or string. Start date in UTC Defaults to 2010-07-06T12:00:00Z (the date the OpenStack project was started)
- **end** `datetime.datetime` or string. End date in UTC. Defaults to now

Returns
A *Usage* object

Raises
SDKException if its not a valid project

get_container(*name*, *skip_cache=False*)

Get metadata about a container.

Parameters

- **name** (*str*) Name of the container to get metadata for.
- **skip_cache** (*bool*) Ignored. Present for backwards compatibility.

Returns
An object store `Container` object if found, else None.

get_container_access(*name*)

Get the control list from a container.

Parameters
name (*str*) Name of the container.

Returns
The contol list for the container.

Raises
SDKException if the container was not found or container access could not be determined.

get_default_network()

Return the network that is configured to be the default interface.

Returns
A network `Network` object if one is found

get_domain(*domain_id=None*, *name_or_id=None*, *filters=None*)

Get exactly one Keystone domain.

Parameters

- **domain_id** ID of the domain.
- **name_or_id** Name or unique ID of the domain.
- **filters DEPRECATED** A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR

A string containing a jmespath expression for further filtering. Example:

```
"[?last_name==`Smith`] | [?other.gender]==`Female`]"
```

Returns
an identity `Domain` object

Raises
SDKException if something goes wrong during the OpenStack API call.

get_endpoint(*id*, *filters=None*)

Get exactly one Keystone endpoint.

Parameters
id ID of endpoint.

Returns
An identity `Endpoint` object

get_external_ipv4_floating_networks()

Return the networks that are configured to route northbound.

Returns
A list of network `Network` objects if any are found

get_external_ipv4_networks()

Return the networks that are configured to route northbound.

Returns
A list of network `Network` objects if any are found

get_external_ipv6_networks()

Return the networks that are configured to route northbound.

Returns
A list of network `Network` objects if any are found

get_external_networks()

Return the networks that are configured to route northbound.

This should be avoided in favor of the specific ipv4/ipv6 method, but is here for backwards compatibility.

Returns
A list of network `Network` objects if any are found

get_firewall_group(*name_or_id*, *filters=None*)

Retrieves firewall group.

Parameters

- **name_or_id** firewall group name or id
- **filters** (*dict*) optional filters

Raises
DuplicateResource on multiple matches

Returns
A network `FirewallGroup` object if found, else None.

get_firewall_policy(*name_or_id*, *filters=None*)

Retrieves a single firewall policy.

Parameters

- **name_or_id** firewall policy name or id
- **filters** A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR A string containing a jmespath expression for further filtering. Example:: [?last_name==‘Smith‘] | [?other.gender]==‘Female‘]

Raises

DuplicateResource on multiple matches

Returns

A network `FirewallPolicy` object if found, else None.

get_firewall_rule(*name_or_id*, *filters=None*)

Retrieves a single firewall rule.

Parameters

- **name_or_id** firewall rule name or id
- **filters** A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR A string containing a jmespath expression for further filtering. Example:: [?last_name==‘Smith‘] | [?other.gender]==‘Female‘]

Raises

DuplicateResource on multiple matches

Returns

A network `FirewallRule` object if found, else None.

get_flavor(*name_or_id*, *filters=None*, *get_extra=True*)

Get a flavor by name or ID.

Parameters

- **name_or_id** Name or ID of the flavor.
- **filters DEPRECATED** A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR A string containing a jmespath expression for further filtering. Example:: [?last_name==‘Smith‘] | [?other.gender]==‘Female‘]

- **get_extra** Whether or not the list_flavors call should get the extra flavor specs.

Returns
A compute `Flavor` object if found, else None.

`get_flavor_by_id`(*id*, *get_extra=False*)

Get a flavor by ID

Parameters

- **id** ID of the flavor.
- **get_extra** Whether or not the list_flavors call should get the extra flavor specs.

Returns
A compute `Flavor` object if found, else None.

`get_flavor_by_ram`(*ram*, *include=None*, *get_extra=True*)

Get a flavor based on amount of RAM available.

Finds the flavor with the least amount of RAM that is at least as much as the specified amount. If *include* is given, further filter based on matching flavor name.

Parameters

- **ram** (*int*) Minimum amount of RAM.
- **include** (*string*) If given, will return a flavor whose name contains this string as a substring.
- **get_extra** Whether to fetch extra specs.

Returns
A compute `Flavor` object.

Raises
SDKException if no matching flavour could be found.

`get_flavor_name`(*flavor_id*)

Get the name of a flavor.

Parameters
flavor_id ID of the flavor.

Returns
The name of the flavor if a match if found, else None.

`get_floating_ip`(*id*, *filters=None*)

Get a floating IP by ID

Parameters

- **id** ID of the floating IP.
- **filters** A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR A string containing a jmespath expression for further filtering. Example:: [?last_name==‘Smith‘] | [?other.gender]==‘Female‘]

Returns
A floating IP `openstack.network.v2.floating_ip.FloatingIP` or None if no matching floating IP is found.

`get_floating_ip_by_id`(*id*)

Get a floating ip by ID

Parameters
id ID of the floating ip.

Returns
A floating ip *:class:'~openstack.network.v2.floating_ip.FloatingIP*.

`get_group`(*name_or_id*, *filters=None*, *domain_id=None*)

Get exactly one Keystone group.

Parameters

- **name_or_id** Name or unique ID of the group(s).
- **domain_id** Domain ID to scope the retrieved group.

Returns
An identity `Group` object

Raises
SDKException if something goes wrong during the OpenStack API call.

`get_image`(*name_or_id*, *filters=None*)

Get an image by name or ID.

Parameters

- **name_or_id** Name or ID of the image.
- **filters** A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR A string containing a jmespath expression for further filtering. Example:: [?last_name=='Smith'] | [?other.gender]=='Female']

Returns
An image *openstack.image.v2.image.Image* object.

`get_image_by_id`(*id*)

Get a image by ID

Parameters
id ID of the image.

Returns
An image *openstack.image.v2.image.Image* object.

`get_internal_ipv4_networks`()

Return the networks that are configured to not route northbound.

Returns
A list of network `Network` objects if any are found

`get_internal_ipv6_networks`()

Return the networks that are configured to not route northbound.

Returns
A list of network `Network` objects if any are found

`get_internal_networks`()

Return the networks that are configured to not route northbound.

This should be avoided in favor of the specific ipv4/ipv6 method, but is here for backwards compatibility.

Returns
A list of network `Network` objects if any are found

`get_keypair`(*name_or_id*, *filters=None*, ***, *user_id=None*)

Get a keypair by name or ID.

Parameters

- **name_or_id** Name or ID of the keypair.
- **filters DEPRECATED** A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR A string containing a jmespath expression for further filtering. Example:: [?last_name==‘Smith‘] | [?other.gender]==‘Female‘]

- **user_id** User to retrieve keypair from.

Returns
A compute `Keypair` object if found, else None.

`get_machine`(*name_or_id*)

Get Machine by name or uuid

Search the baremetal host out by utilizing the supplied id value which can consist of a name or UUID.

Parameters
name_or_id A node name or UUID that will be looked up.

Return type
Node.

Returns
The node found or None if no nodes are found.

`get_machine_by_mac`(*mac*)

Get machine by port MAC address

Parameters
mac Port MAC address to query in order to return a node.

Return type
Node.

Returns
The node found or None if no nodes are found.

`get_nat_destination()`

Return the network that is configured to be the NAT destination.

Returns
A network `Network` object if one is found

`get_nat_source()`

Return the network that is configured to be the NAT destination.

Returns
A network `Network` object if one is found

`get_network(name_or_id, filters=None)`

Get a network by name or ID.

Parameters

- **`name_or_id`** Name or ID of the network.
- **`filters`** A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR A string containing a jmespath expression for further filtering. Example:: [?last_name==‘Smith‘] | [?other.gender]==‘Female‘]

Returns
A network `Network` object if found, else None.

`get_network_by_id(id)`

Get a network by ID

Parameters
`id` ID of the network.

Returns
A network `Network` object if found, else None.

`get_network_extensions()`

Get Cloud provided network extensions

Returns
A set of Neutron extension aliases.

`get_network_quotas(name_or_id, details=False)`

Get network quotas for a project

Parameters

- **`name_or_id`** project name or id
- **`details`** if set to True it will return details about usage of quotas by given project

Returns
A network `Quota` object if found, else None.

Raises
SDKException if its not a valid project

get_nic_by_mac(*mac*)

Get bare metal NIC by its hardware address (usually MAC).

get_object(*container*, *obj*, *query_string=None*, *resp_chunk_size=1024*, *outfile=None*, *stream=False*)

Get the headers and body of an object

Parameters

- **container** (*string*) Name of the container.
- **obj** (*string*) Name of the object.
- **query_string** (*string*) Query args for uri. (delimiter, prefix, etc.)
- **resp_chunk_size** (*int*) Chunk size of data to read. Only used if the results are being written to a file or stream is True. (optional, defaults to 1k)
- **outfile** Write the object to a file instead of returning the contents. If this option is given, body in the return tuple will be None. outfile can either be a file path given as a string, or a File like object.

Returns
Tuple (headers, body) of the object, or None if the object is not found (404).

Raises
SDKException on operation error.

get_object_capabilities()

Get infomation about the object-storage service

The object-storage service publishes a set of capabilities that include metadata about maximum values and thresholds.

Returns
An object store `Info` object.

get_object_metadata(*container*, *name*)

Get object metadata.

Parameters

- **container**
- **name**

Returns
The object metadata.

get_object_raw(*container*, *obj*, *query_string=None*, *stream=False*)

Get a raw response object for an object.

Parameters

- **container** (*string*) Name of the container.
- **obj** (*string*) Name of the object.
- **query_string** (*string*) Query args for uri. (delimiter, prefix, etc.)

- **stream** (*bool*) Whether to stream the response or not.

Returns
A *requests.Response*

Raises
SDKException on operation error.

get_object_segment_size(*segment_size*)

Get a segment size that will work given capabilities.

Parameters
segment_size

Returns
A segment size.

get_port(*name_or_id*, *filters=None*)

Get a port by name or ID.

Parameters

- **name_or_id** Name or ID of the port.
- **filters** A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR A string containing a jmespath expression for further filtering. Example:: [?last_name==‘Smith‘] | [?other.gender]==‘Female‘]

Returns
A network `Port` object if found, else None.

get_port_by_id(*id*)

Get a port by ID

Parameters
id ID of the port.

Returns
A network `Port` object if found, else None.

get_project(*name_or_id*, *filters=None*, *domain_id=None*)

Get exactly one project.

Parameters

- **name_or_id** Name or unique ID of the project.
- **filters DEPRECATED** A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR

A string containing a jmespath expression for further filtering. Example:

```
"[?last_name==`Smith`] | [?other.gender]==`Female`]"
```

- **domain_id** Domain ID to scope the retrieved project.

Returns
An identity `Project` object.

Raises
SDKException if something goes wrong during the OpenStack API call.

get_qos_bandwidth_limit_rule(*policy_name_or_id*, *rule_id*)

Get a QoS bandwidth limit rule by name or ID.

Parameters

- **policy_name_or_id** (*string*) Name or ID of the QoS policy to which rule should be associated.
- **rule_id** ID of the rule.

Returns
A network `QoSBandwidthLimitRule` object if found, else None.

get_qos_dscp_marking_rule(*policy_name_or_id*, *rule_id*)

Get a QoS DSCP marking rule by name or ID.

Parameters

- **policy_name_or_id** (*string*) Name or ID of the QoS policy to which rule should be associated.
- **rule_id** ID of the rule.

Returns
A network `QoSDSCPMarkingRule` object if found, else None.

get_qos_minimum_bandwidth_rule(*policy_name_or_id*, *rule_id*)

Get a QoS minimum bandwidth rule by name or ID.

Parameters

- **policy_name_or_id** (*string*) Name or ID of the QoS policy to which rule should be associated.
- **rule_id** ID of the rule.

Returns
A network `QoSMinimumBandwidthRule` object if found, else None.

get_qos_policy(*name_or_id*, *filters=None*)

Get a QoS policy by name or ID.

Parameters

- **name_or_id** Name or ID of the policy.
- **filters** A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR A string containing a jmespath expression for further filtering. Example:: [?last_name==‘Smith‘] | [?other.gender]==‘Female‘]

Returns
A network `QoSPolicy` object if found, else None.

get_qos_rule_type_details(*rule_type*, *filters=None*)

Get a QoS rule type details by rule type name.

Parameters

- **rule_type** Name of the QoS rule type.
- **filters** A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR A string containing a jmespath expression for further filtering. Example:: [?last_name==‘Smith‘] | [?other.gender]==‘Female‘]

Returns
A network `QoSRuleType` object if found, else None.

get_recordset(*zone*, *name_or_id*)

Get a recordset by name or ID.

Parameters

- **zone** Name, ID or *openstack.dns.v2.zone.Zone* instance of the zone managing the recordset.
- **name_or_id** Name or ID of the recordset

Returns
A recordset dict or None if no matching recordset is found.

get_role(*name_or_id*, *filters=None*, *domain_id=None*)

Get a Keystone role.

Parameters

- **name_or_id** Name or unique ID of the role.
- **domain_id** Domain ID to scope the retrieved role.

Returns
An identity `Role` object if found, else None.

Raises
SDKException if something goes wrong during the OpenStack API call.

get_router(*name_or_id*, *filters=None*)

Get a router by name or ID.

Parameters

- **name_or_id** Name or ID of the router.

- **filters** A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

  OR A string containing a jmespath expression for further filtering. Example:: [?last_name==‘Smith‘] | [?other.gender]==‘Female‘]

Returns
A network `Router` object if found, else None.

get_security_group(*name_or_id*, *filters=None*)

Get a security group by name or ID.

Parameters

- **name_or_id** Name or ID of the security group.
- **filters** A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

  OR A string containing a jmespath expression for further filtering. Example:: [?last_name==‘Smith‘] | [?other.gender]==‘Female‘]

Returns
A security group `openstack.network.v2.security_group.SecurityGroup` or None if no matching security group is found.

get_security_group_by_id(*id*)

Get a security group by ID

Parameters
id ID of the security group.

Returns
A security group `openstack.network.v2.security_group.SecurityGroup`.

get_server(*name_or_id=None*, *filters=None*, *detailed=False*, *bare=False*, *all_projects=False*)

Get a server by name or ID.

Parameters

- **name_or_id** Name or ID of the server.
- **filters DEPRECATED** A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

  OR A string containing a jmespath expression for further filtering. Example:: [?last_name==‘Smith‘] | [?other.gender]==‘Female‘]

- **detailed** Whether or not to add detailed additional information. Defaults to False.

- **bare** Whether to skip adding any additional information to the server record. Defaults to False, meaning the addresses dict will be populated as needed from neutron. Setting to True implies detailed = False.
- **all_projects** Whether to get server from all projects or just the current auth scoped project.

Returns
A compute `Server` object if found, else None.

`get_server_by_id`(*id*)

Get a server by ID.

Parameters
id ID of the server.

Returns
A compute `Server` object if found, else None.

`get_server_console`(*server*, *length=None*)

Get the console log for a server.

Parameters

- **server** The server to fetch the console log for. Can be either a server dict or the Name or ID of the server.
- **length** (*int*) The number of lines you would like to retrieve from the end of the log. (optional, defaults to all)

Returns
A string containing the text of the console log or an empty string if the cloud does not support console logs.

Raises
SDKException if an invalid server argument is given or if something else unforseen happens

`get_server_group`(*name_or_id=None*, *filters=None*)

Get a server group by name or ID.

Parameters

- **name_or_id** Name or ID of the server group.
- **filters DEPRECATED** A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{
    'policy': 'affinity',
}
```

OR A string containing a jmespath expression for further filtering. Example:: [?last_name==‘Smith‘] | [?other.gender]==‘Female‘]

Returns
A compute `ServerGroup` object if found, else None.

get_server_id(*name_or_id*)

Get the ID of a server.

Parameters
name_or_id

Returns
The name of the server if found, else None.

get_server_meta(*server*)

Get the metadata for a server.

Parameters
server

Returns
The metadata for the server if found, else None.

get_server_private_ip(*server*)

Get the private IP of a server.

Parameters
server

Returns
The private IP of the server if set, else None.

get_server_public_ip(*server*)

Get the public IP of a server.

Parameters
server

Returns
The public IP of the server if set, else None.

get_service(*name_or_id*, *filters=None*)

Get exactly one Keystone service.

Parameters
name_or_id Name or unique ID of the service.

Returns
an identity `Service` object

Raises
SDKException if something goes wrong during the OpenStack API call or if multiple matches are found.

get_stack(*name_or_id*, *filters=None*, *resolve_outputs=True*)

Get exactly one stack.

Parameters

- **name_or_id** Name or ID of the desired stack.
- **filters** a dict containing additional filters to use. e.g. {stack_status: CREATE_COMPLETE}
- **resolve_outputs** If True, then outputs for this stack will be resolved

Returns
a *openstack.orchestration.v1.stack.Stack* containing the stack description

Raises
SDKException if something goes wrong during the OpenStack API call or if multiple matches are found.

get_subnet(*name_or_id*, *filters=None*)

Get a subnet by name or ID.

Parameters

- **name_or_id** Name or ID of the subnet.
- **filters** A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

Returns
A network `Subnet` object if found, else None.

get_subnet_by_id(*id*)

Get a subnet by ID

Parameters
id ID of the subnet.

Returns
A network `Subnet` object if found, else None.

get_subnetpool(*name_or_id*)

Get a subnetpool by name or ID.

Parameters
name_or_id Name or ID of the subnetpool.

Returns
A network `Subnetpool` object if found, else None.

get_user(*name_or_id*, *filters=None*, *domain_id=None*)

Get exactly one user.

Parameters

- **name_or_id** Name or unique ID of the user.
- **filters DEPRECATED** A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR

A string containing a jmespath expression for further filtering. Example:

```
"[?last_name==`Smith`] | [?other.gender]==`Female`]"
```

- **domain_id** Domain ID to scope the retrieved user.

Returns
an identity `User` object

Raises
SDKException if something goes wrong during the OpenStack API call.

get_user_by_id(*user_id*, *normalize=None*)

Get a user by ID.

Parameters
user_id (*string*) user ID

Returns
an identity `User` object

get_volume(*name_or_id*, *filters=None*)

Get a volume by name or ID.

Parameters

- **name_or_id** Name or unique ID of the volume.
- **filters DEPRECATED** A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR

A string containing a jmespath expression for further filtering. Example:

```
"[?last_name==`Smith`] | [?other.gender]==`Female`]"
```

Returns
A volume `Volume` object if found, else None.

get_volume_attach_device(*volume*, *server_id*)

Return the device name a volume is attached to for a server.

This can also be used to verify if a volume is attached to a particular server.

Parameters

- **volume** The volume to fetch the device name from.
- **server_id** ID of server to check.

Returns
Device name if attached, None if volume is not attached.

get_volume_backup(*name_or_id*, *filters=None*)

Get a volume backup by name or ID.

Parameters

- **name_or_id** Name or unique ID of the volume backup.
- **filters DEPRECATED** A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR

A string containing a jmespath expression for further filtering. Example:

```
"[?last_name==`Smith`] | [?other.gender]==`Female`]"
```

Returns
A volume `Backup` object if found, else None.

`get_volume_by_id`(*id*)

Get a volume by ID

Parameters
id ID of the volume.

Returns
A volume `Volume` object if found, else None.

`get_volume_id`(*name_or_id*)

Get ID of a volume.

Parameters
name_or_id Name or unique ID of the volume.

Returns
The ID of the volume if found, else None.

`get_volume_limits`(*name_or_id=None*)

Get volume limits for the current project

Parameters
name_or_id (optional) Project name or ID to get limits for if different from the current project

Returns
The volume `Limits` object if found, else None.

`get_volume_quotas`(*name_or_id*)

Get volume quotas for a project

Parameters
name_or_id project name or id

Returns
A volume `QuotaSet` object with the quotas

Raises
SDKException if its not a valid project

`get_volume_snapshot`(*name_or_id*, *filters=None*)

Get a volume by name or ID.

Parameters

- **name_or_id** Name or unique ID of the volume snapshot.

- **filters DEPRECATED** A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR

A string containing a jmespath expression for further filtering. Example:

```
"[?last_name==`Smith`] | [?other.gender]==`Female`]"
```

Returns
A volume Snapshot object if found, else None.

get_volume_snapshot_by_id(*snapshot_id*)

Takes a snapshot_id and gets a dict of the snapshot that maches that ID.

Note: This is more efficient than get_volume_snapshot.

param: snapshot_id: ID of the volume snapshot. :returns: A volume Snapshot object if found, else None.

get_volume_type(*name_or_id*, *filters=None*)

Get a volume type by name or ID.

Parameters

- **name_or_id** Name or unique ID of the volume type.
- **filters DEPRECATED** A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR

A string containing a jmespath expression for further filtering. Example:

```
"[?last_name==`Smith`] | [?other.gender]==`Female`]"
```

Returns
A volume Type object if found, else None.

get_volume_type_access(*name_or_id*)

Return a list of volume_type_access.

Parameters
name_or_id Name or unique ID of the volume type.

Returns
A volume Type object if found, else None.

Raises
SDKException on operation error.

get_volumes(*server*, *cache=None*)

Get volumes for a server.

Parameters

- **server** The server to fetch volumes for.
- **cache** **DEPRECATED** This parameter no longer does anything.

Returns
A list of volume `Volume` objects.

get_zone(*name_or_id*, *filters=None*)

Get a zone by name or ID.

Parameters

- **name_or_id** Name or ID of the zone
- **filters** A dictionary of meta data to use for further filtering

Returns
A zone dict or None if no matching zone is found.

global_request(*global_request_id*)

Make a new Connection object with a global request id set.

Take the existing settings from the current Connection and construct a new Connection object with the global_request_id overridden.

```
from oslo_context import context

cloud = openstack.connect(cloud='example')
# Work normally
servers = cloud.list_servers()
cloud2 = cloud.global_request(context.generate_request_id())
# cloud2 sends all requests with global_request_id set
servers = cloud2.list_servers()
```

Additionally, this can be used as a context manager:

```
from oslo_context import context

c = openstack.connect(cloud='example')
# Work normally
servers = c.list_servers()
with c.global_request(context.generate_request_id()) as c2:
    # c2 sends all requests with global_request_id set
    servers = c2.list_servers()
```

Parameters
global_request_id The *global_request_id* to send.

grant_role(*name_or_id*, *user=None*, *group=None*, *project=None*, *domain=None*, *system=None*, *inherited=False*, *wait=False*, *timeout=60*)

Grant a role to a user.

Parameters

- **name_or_id** (*string*) Name or unique ID of the role.
- **user** (*string*) The name or id of the user.

- **group** (*string*) The name or id of the group. (v3)
- **project** (*string*) The name or id of the project.
- **domain** (*string*) The id of the domain. (v3)
- **system** (*bool*) The name of the system. (v3)
- **inherited** (*bool*) Whether the role assignment is inherited. (v3)
- **wait** (*bool*) Wait for role to be granted
- **timeout** (*int*) Timeout to wait for role to be granted

  NOTE: domain is a required argument when the grant is on a project, user or group specified by name. In that situation, they are all considered to be in that domain. If different domains are in use in the same role grant, it is required to specify those by ID.

  NOTE: for wait and timeout, sometimes granting roles is not instantaneous.

NOTE: precedence is given first to project, then domain, then system

Returns
True if the role is assigned, otherwise False

Raises
SDKException if the role cannot be granted

insert_rule_into_policy(*name_or_id*, *rule_name_or_id*, *insert_after=None*, *insert_before=None*, *filters=None*)

Add firewall rule to a policy.

Adds firewall rule to the firewall_rules list of a firewall policy. Short-circuits and returns the firewall policy early if the firewall rule id is already present in the firewall_rules list. This method doesnt do re-ordering. If you want to move a firewall rule or down the list, you have to remove and re-add it.

Parameters

- **name_or_id** firewall policy name or id
- **rule_name_or_id** firewall rule name or id
- **insert_after** rule name or id that should precede added rule
- **insert_before** rule name or id that should succeed added rule
- **filters** (*dict*) optional filters

Raises
DuplicateResource on multiple matches

Raises
NotFoundException if firewall policy or any of the firewall rules (inserted, after, before) is not found.

Returns
updated firewall policy

Return type
FirewallPolicy

inspect_machine(*name_or_id*, *wait=False*, *timeout=3600*)

Inspect a Barmetal machine

Engages the Ironic node inspection behavior in order to collect metadata about the baremetal machine.

Parameters

- **name_or_id** String representing machine name or UUID value in order to identify the machine.
- **wait** Boolean value controlling if the method is to wait for the desired state to be reached or a failure to occur.
- **timeout** Integer value, defautling to 3600 seconds, for the wait state to reach completion.

Return type
Node.

Returns
Current state of the node.

is_object_stale(*container*, *name*, *filename*, *file_md5=None*, *file_sha256=None*)

Check to see if an object matches the hashes of a file.

Parameters

- **container** Name of the container.
- **name** Name of the object.
- **filename** Path to the file.
- **file_md5** Pre-calculated md5 of the file contents. Defaults to None which means calculate locally.
- **file_sha256** Pre-calculated sha256 of the file contents. Defaults to None which means calculate locally.

is_user_in_group(*name_or_id*, *group_name_or_id*)

Check to see if a user is in a group.

Parameters

- **name_or_id** Name or unique ID of the user.
- **group_name_or_id** Group name or ID

Returns
True if user is in the group, False otherwise

Raises
SDKException if something goes wrong during the OpenStack API call

list_accelerator_requests(*filters=None*)

List all accelerator_requests.

Parameters
filters (optional) dict of filter conditions to push down

Returns
A list of accelerator `AcceleratorRequest` objects.

`list_aggregates`(*filters=None*)

List all available host aggregates.

Returns
A list of compute `Aggregate` objects.

`list_availability_zone_names`(*unavailable=False*)

List names of availability zones.

Parameters
`unavailable` (*`bool`*) Whether or not to include unavailable zones in the output. Defaults to False.

Returns
A list of availability zone names, or an empty list if the list could not be fetched.

`list_cluster_templates`(*detail=False*)

List cluster templates.

:param bool detail. Ignored. Included for backwards compat.
ClusterTemplates are always returned with full details.

Returns
a list of dicts containing the cluster template details.

Raises
`SDKException` if something goes wrong during the OpenStack API call.

`list_coe_clusters`()

List COE (Container Orchestration Engine) cluster.

Returns
A list of container infrastructure management `Cluster` objects.

Raises
`SDKException` if something goes wrong during the OpenStack API call.

`list_containers`(*full_listing=None*, *prefix=None*)

List containers.

Parameters

- **`full_listing`** Ignored. Present for backwards compat
- **`prefix`** Only objects with this prefix will be returned. (optional)

Returns
A list of object store `Container` objects.

Raises
`SDKException` on operation error.

`list_deployables`(*filters=None*)

List all available deployables.

Parameters
filters (optional) dict of filter conditions to push down

Returns
A list of accelerator `Deployable` objects.

`list_device_profiles`(*filters=None*)

List all device_profiles.

Parameters
filters (optional) dict of filter conditions to push down

Returns
A list of accelerator `DeviceProfile` objects.

`list_devices`(*filters=None*)

List all devices.

Parameters
filters (optional) dict of filter conditions to push down

Returns
A list of accelerator `Device` objects.

`list_domains`(***filters*)

List Keystone domains.

Returns
A list of identity `Domain` objects.

Raises
SDKException if something goes wrong during the OpenStack API call.

`list_endpoints`()

List Keystone endpoints.

Returns
A list of identity `Endpoint` objects

Raises
SDKException if something goes wrong during the OpenStack API call.

`list_firewall_groups`(*filters=None*)

Lists firewall groups.

Returns
A list of network `FirewallGroup` objects.

`list_firewall_policies`(*filters=None*)

Lists firewall policies.

Parameters
filters A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR A string containing a jmespath expression for further filtering. Example:: [?last_name==‘Smith‘] | [?other.gender]==‘Female‘]

Returns
A list of network `FirewallPolicy` objects.

Return type
list[FirewallPolicy]

`list_firewall_rules`(*filters=None*)

Lists firewall rules.

Parameters
`filters` A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR A string containing a jmespath expression for further filtering. Example:: [?last_name==‘Smith‘] | [?other.gender]==‘Female‘]

Returns
A list of network `FirewallRule` objects.

Return type
list[FirewallRule]

`list_flavor_access`(*flavor_id*)

List access from a private flavor for a project/tenant.

Parameters
`flavor_id` (*string*) ID of the private flavor.

Returns
List of dicts with flavor_id and tenant_id attributes.

Raises
SDKException on operation error.

`list_flavors`(*get_extra=False*)

List all available flavors.

Parameters
`get_extra` Whether or not to fetch extra specs for each flavor. Defaults to True. Default behavior value can be overridden in clouds.yaml by setting openstack.cloud.get_extra_specs to False.

Returns
A list of compute `Flavor` objects.

`list_floating_ip_pools`()

List all available floating IP pools.

NOTE: This function supports the nova-net view of the world. nova-net has been deprecated, so its highly recommended to switch to using neutron. *get_external_ipv4_floating_networks* is what you should almost certainly be using.

Returns
A list of floating IP pool objects

`list_floating_ips`(*filters=None*)

List all available floating IPs.

Parameters
filters (optional) dict of filter conditions to push down

Returns
A list of floating IP `openstack.network.v2.floating_ip.FloatingIP`.

`list_groups`(***kwargs*)

List Keystone groups.

Parameters
domain_id Domain ID.

Returns
A list of identity `Group` objects

Raises
SDKException if something goes wrong during the OpenStack API call.

`list_hypervisors`(*filters=None*)

List all hypervisors

Parameters
filters Additional query parameters passed to the API server.

Returns
A list of compute `Hypervisor` objects.

`list_images`(*filter_deleted=True*, *show_all=False*)

Get available images.

Parameters

- **filter_deleted** Control whether deleted images are returned.
- **show_all** Show all images, including images that are shared but not accepted. (By default in glance v2 shared image that have not been accepted are not shown) show_all will override the value of filter_deleted to False.

Returns
A list of glance images.

`list_keypairs`(*filters=None*)

List all available keypairs.

Parameters
filters

Returns
A list of compute `Keypair` objects.

`list_machines`()

List Machines.

Returns
list of *Node*.

list_magnum_services()

List all Magnum services.

Returns
a list of dicts containing the service details.

Raises
SDKException on operation error.

list_networks(*filters=None*)

List all available networks.

Parameters
filters (optional) A dict of filter conditions to push down.

Returns
A list of network `Network` objects.

list_nics()

Return a list of all bare metal ports.

list_nics_for_machine(*uuid*)

Returns a list of ports present on the machine node.

Parameters
uuid String representing machine UUID value in order to identify the machine.

Returns
A list of ports.

list_objects(*container*, *full_listing=True*, *prefix=None*)

List objects.

Parameters

- **container** Name of the container to list objects in.
- **full_listing** Ignored. Present for backwards compat
- **prefix** Only objects with this prefix will be returned. (optional)

Returns
A list of object store `Object` objects.

Raises
SDKException on operation error.

list_ports(*filters=None*)

List all available ports.

Parameters
filters (optional) A dict of filter conditions to push down

Returns
A list of network `Port` objects.

list_ports_attached_to_machine(*name_or_id*)

List virtual ports attached to the bare metal machine.

Parameters
name_or_id (*string*) A machine name or UUID.

Returns
List of `openstack.Resource` objects representing the ports.

`list_projects`(*domain_id=None*, *name_or_id=None*, *filters=None*)

List projects.

With no parameters, returns a full listing of all visible projects.

Parameters

- **domain_id** Domain ID to scope the searched projects.
- **name_or_id** Name or ID of the project(s).
- **filters** A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR

A string containing a jmespath expression for further filtering. Example:

```
"[?last_name==`Smith`] | [?other.gender]==`Female`]"
```

Returns
A list of identity `Project` objects.

Raises
SDKException if something goes wrong during the OpenStack API call.

`list_qos_bandwidth_limit_rules`(*policy_name_or_id*, *filters=None*)

List all available QoS bandwidth limit rules.

Parameters

- **policy_name_or_id** (*string*) Name or ID of the QoS policy from from rules should be listed.
- **filters** (optional) A dict of filter conditions to push down

Returns
A list of network `QoSBandwidthLimitRule` objects.

Raises
`:class:`~openstack.exceptions.BadRequestException`` if QoS policy will not be found.

`list_qos_dscp_marking_rules`(*policy_name_or_id*, *filters=None*)

List all available QoS DSCP marking rules.

Parameters

- **policy_name_or_id** (*string*) Name or ID of the QoS policy from from rules should be listed.
- **filters** (optional) A dict of filter conditions to push down

Returns
A list of network `QoSDSCPMarkingRule` objects.

Raises
`:class:`~openstack.exceptions.BadRequestException`` if QoS policy will not be found.

`list_qos_minimum_bandwidth_rules`(*policy_name_or_id*, *filters=None*)

List all available QoS minimum bandwidth rules.

Parameters

- **policy_name_or_id** (*string*) Name or ID of the QoS policy from from rules should be listed.
- **filters** (optional) A dict of filter conditions to push down

Returns
A list of network `QoSMinimumBandwidthRule` objects.

Raises
`:class:`~openstack.exceptions.BadRequestException`` if QoS policy will not be found.

`list_qos_policies`(*filters=None*)

List all available QoS policies.

Parameters
filters (optional) A dict of filter conditions to push down

Returns
A list of network `QosPolicy` objects.

`list_qos_rule_types`(*filters=None*)

List all available QoS rule types.

Parameters
filters (optional) A dict of filter conditions to push down

Returns
A list of network `QosRuleType` objects.

`list_recordsets`(*zone*)

List all available recordsets.

Parameters
zone Name, ID or *openstack.dns.v2.zone.Zone* instance of the zone managing the recordset.

Returns
A list of recordsets.

`list_role_assignments`(*filters=None*)

List Keystone role assignments

Parameters
filters (*dict*) Dict of filter conditions. Acceptable keys are:

- user (string) - User ID to be used as query filter.

- group (string) - Group ID to be used as query filter.
- project (string) - Project ID to be used as query filter.
- domain (string) - Domain ID to be used as query filter.
- system (string) - System name to be used as query filter.
- role (string) - Role ID to be used as query filter.
- inherited_to (string) - Return inherited role assignments for either projects or domains.
- os_inherit_extension_inherited_to (string) - Deprecated; use inherited_to instead.
- effective (boolean) - Return effective role assignments.
- include_subtree (boolean) - Include subtree

  user and group are mutually exclusive, as are domain and project.

Returns
A list of identity *openstack.identity.v3.role_assignment.RoleAssignment* objects

Raises
SDKException if something goes wrong during the OpenStack API call.

list_roles(***kwargs*)

List Keystone roles.

Returns
A list of identity `Role` objects

Raises
SDKException if something goes wrong during the OpenStack API call.

list_router_interfaces(*router*, *interface_type=None*)

List all interfaces for a router.

Parameters
- **router** (*dict*) A router dict object.
- **interface_type** (*string*) One of None, internal, or external. Controls whether all, internal interfaces or external interfaces are returned.

Returns
A list of network `Port` objects.

list_routers(*filters=None*)

List all available routers.

Parameters
filters (optional) A dict of filter conditions to push down

Returns
A list of network `Router` objects.

list_security_groups(*filters=None*)

List all available security groups.

Parameters

filters (optional) dict of filter conditions to push down

Returns

A list of security group `openstack.network.v2.security_group.SecurityGroup`.

list_server_groups()

List all available server groups.

Returns

A list of compute `ServerGroup` objects.

list_server_security_groups(*server*)

List all security groups associated with the given server.

Returns

A list of security group dictionary objects.

list_servers(*detailed=False*, *all_projects=False*, *bare=False*, *filters=None*)

List all available servers.

Parameters

- **detailed** Whether or not to add detailed additional information. Defaults to False.
- **all_projects** Whether to list servers from all projects or just the current auth scoped project.
- **bare** Whether to skip adding any additional information to the server record. Defaults to False, meaning the addresses dict will be populated as needed from neutron. Setting to True implies detailed = False.
- **filters** Additional query parameters passed to the API server.

Returns

A list of compute `Server` objects.

list_services()

List all Keystone services.

Returns

A list of identity `Service` object

Raises

SDKException if something goes wrong during the OpenStack API call.

list_share_availability_zones()

List all availability zones for the Shared File Systems service.

Returns

A list of Shared File Systems Availability Zones.

list_stacks(***query*)

List all stacks.

Parameters
query (*dict*) Query parameters to limit stacks.

Returns
a list of *openstack.orchestration.v1.stack.Stack* objects containing the stack description.

Raises
SDKException if something goes wrong during the OpenStack API call.

list_subnets(*filters=None*)

List all available subnets.

Parameters
filters (optional) A dict of filter conditions to push down

Returns
A list of network `Subnet` objects.

list_users(***kwargs*)

List users.

Parameters

- **name**
- **domain_id** Domain ID to scope the retrieved users.

Returns
A list of identity `User` objects.

Raises
SDKException if something goes wrong during the OpenStack API call.

list_volume_backups(*detailed=True*, *filters=None*)

List all volume backups.

Parameters

- **detailed** Whether or not to add detailed additional information.
- **filters** A dictionary of meta data to use for further filtering. Example:

```
{
    'name': 'my-volume-backup',
    'status': 'available',
    'volume_id': 'e126044c-7b4c-43be-a32a-c9cbbc9ddb56',
    'all_tenants': 1,
}
```

Returns
A list of volume `Backup` objects.

list_volume_snapshots(*detailed=True*, *filters=None*)

List all volume snapshots.

Parameters

- **detailed** Whether or not to add detailed additional information.

- **filters** A dictionary of meta data to use for further filtering. Example:

```
{
    'name': 'my-volume-snapshot',
    'volume_id': 'e126044c-7b4c-43be-a32a-c9cbbc9ddb56',
    'all_tenants': 1,
}
```

Returns
A list of volume `Snapshot` objects.

`list_volume_types`(*get_extra=None*)

List all available volume types.

Parameters
get_extra **DEPRECATED** This parameter no longer does anything.

Returns
A list of volume `Type` objects.

`list_volumes`(*cache=None*)

List all available volumes.

Parameters
cache **DEPRECATED** This parameter no longer does anything.

Returns
A list of volume `Volume` objects.

`list_zones`(*filters=None*)

List all available zones.

Returns
A list of zones dicts.

`node_set_provision_state`(*name_or_id*, *state*, *configdrive=None*, *wait=False*, *timeout=3600*)

Set Node Provision State

Enables a user to provision a Machine and optionally define a config drive to be utilized.

Parameters

- **name_or_id** (*string*) The Name or UUID value representing the baremetal node.
- **state** (*string*) The desired provision state for the baremetal node.
- **configdrive** (*string*) An optional URL or file or path representing the configdrive. In the case of a directory, the client API will create a properly formatted configuration drive file and post the file contents to the API for deployment.
- **wait** (*boolean*) A boolean value, defaulted to false, to control if the method will wait for the desire end state to be reached before returning.
- **timeout** (*integer*) Integer value, defaulting to 3600 seconds, representing the amount of time to wait for the desire end state to be reached.

Returns
Current state of the machine upon exit of the method.

Return type
Node.

Raises
SDKException on operation error.

patch_machine(*name_or_id*, *patch*)

Patch Machine Information

This method allows for an interface to manipulate node entries within Ironic.

Parameters

- **name_or_id** (*string*) A machine name or UUID to be updated.
- **patch** The JSON Patch document is a list of dictonary objects that comply with RFC 6902 which can be found at https://tools.ietf.org/html/rfc6902.

  Example patch construction:

```
patch = []
patch.append({'op': 'remove', 'path': '/instance_info'})
patch.append(
    {'op': 'replace', 'path': '/name', 'value': 'newname
↪'}
)
patch.append(
    {
        'op': 'add',
        'path': '/driver_info/username',
        'value': 'administrator',
    }
)
```

Returns
Current state of the node.

Return type
Node.

Raises
SDKException on operation error.

pformat(*resource*)

Wrapper around pformat that groks munch objects

pprint(*resource*)

Wrapper around pprint that groks munch objects

project_cleanup(*dry_run=True*, *wait_timeout=120*, *status_queue=None*, *filters=None*, *resource_evaluation_fn=None*, *skip_resources=None*)

Cleanup the project resources.

Cleanup all resources in all services, which provide cleanup methods.

Parameters

- **dry_run** (*bool*) Cleanup or only list identified resources.
- **wait_timeout** (*int*) Maximum amount of time given to each service to comlete the cleanup.
- **status_queue** (*queue*) a threading queue object used to get current process status. The queue contain processed resources.
- **filters** (*dict*) Additional filters for the cleanup (only resources matching all filters will be deleted, if there are no other dependencies).
- **resource_evaluation_fn** A callback function, which will be invoked for each resurce and must return True/False depending on whether resource need to be deleted or not.
- **skip_resources** List of specific resources whose cleanup should be skipped.

range_search(*data*, *filters*)

Perform integer range searches across a list of dictionaries.

Given a list of dictionaries, search across the list using the given dictionary keys and a range of integer values for each key. Only dictionaries that match ALL search filters across the entire original data set will be returned.

It is not a requirement that each dictionary contain the key used for searching. Those without the key will be considered non-matching.

The range values must be string values and is either a set of digits representing an integer for matching, or a range operator followed by a set of digits representing an integer for matching. If a range operator is not given, exact value matching will be used. Valid operators are one of: <,>,<=,>=

Parameters

- **data** List of dictionaries to be searched.
- **filters** Dict describing the one or more range searches to perform. If more than one search is given, the result will be the members of the original data set that match ALL searches. An example of filtering by multiple ranges:

```
{"vcpus": "<=5", "ram": "<=2048", "disk": "1"}
```

Returns

A list subset of the original data set.

Raises

SDKException on invalid range expressions.

rebuild_server(*server_id*, *image_id*, *admin_pass=None*, *detailed=False*, *bare=False*, *wait=False*, *timeout=180*)

Rebuild a server.

Parameters

- **server_id**
- **image_id**

- **admin_pass**
- **detailed**
- **bare**
- **wait**
- **timeout**

Returns
A compute `Server` object.

register_machine(*nics*, *wait=False*, *timeout=3600*, *lock_timeout=600*, *provision_state='available'*, ***kwargs*)

Register Baremetal with Ironic

Allows for the registration of Baremetal nodes with Ironic and population of pertinant node information or configuration to be passed to the Ironic API for the node.

This method also creates ports for a list of MAC addresses passed in to be utilized for boot and potentially network configuration.

If a failure is detected creating the network ports, any ports created are deleted, and the node is removed from Ironic.

Parameters

- **nics** An array of ports that represent the network interfaces for the node to be created. The ports are created after the node is enrolled but before it goes through cleaning.

  Example:

```
[
    {'address': 'aa:bb:cc:dd:ee:01'},
    {'address': 'aa:bb:cc:dd:ee:02'},
]
```

  Alternatively, you can provide an array of MAC addresses.

- **wait** Boolean value, defaulting to false, to wait for the node to reach the available state where the node can be provisioned. It must be noted, when set to false, the method will still wait for locks to clear before sending the next required command.
- **timeout** Integer value, defautling to 3600 seconds, for the wait state to reach completion.
- **lock_timeout** Integer value, defaulting to 600 seconds, for locks to clear.
- **provision_state** The expected provision state, one of enroll manageable or available. Using available results in automated cleaning.
- **kwargs** Key value pairs to be passed to the Ironic API, including uuid, name, chassis_uuid, driver_info, properties.

Returns
Current state of the node.

Return type
Node.

Raises
SDKException on operation error.

remove_flavor_access(*flavor_id*, *project_id*)

Revoke access from a private flavor for a project/tenant.

Parameters

- **flavor_id** (*string*) ID of the private flavor.
- **project_id** (*string*) ID of the project/tenant.

Returns
None

Raises
SDKException on operation error.

remove_host_from_aggregate(*name_or_id*, *host_name*)

Remove a host from an aggregate.

Parameters

- **name_or_id** Name or ID of the host aggregate.
- **host_name** Host to remove.

Raises
SDKException on operation error.

remove_machine_from_maintenance(*name_or_id*)

Remove Baremetal Machine from Maintenance State

Similarly to set_machine_maintenance_state, this method removes a machine from maintenance state. It must be noted that this method simpily calls set_machine_maintenace_state for the name_or_id requested and sets the state to False.

Parameters
name_or_id (*string*) The Name or UUID value representing the baremetal node.

Returns
None

Raises
SDKException on operation error.

remove_router_interface(*router*, *subnet_id=None*, *port_id=None*)

Detach a subnet from an internal router interface.

At least one of subnet_id or port_id must be supplied.

If you specify both subnet and port ID, the subnet ID must correspond to the subnet ID of the first IP address on the port specified by the port ID. Otherwise an error occurs.

Parameters

- **router** (*dict*) The dict object of the router being changed

- **subnet_id** (*string*) The ID of the subnet to use for the interface
- **port_id** (*string*) The ID of the port to use for the interface

Returns
None on success

Raises
SDKException on operation error.

remove_rule_from_policy(*name_or_id*, *rule_name_or_id*, *filters=None*)

Remove firewall rule from firewall policys firewall_rules list. Short-circuits and returns firewall policy early if firewall rule is already absent from the firewall_rules list.

Parameters

- **name_or_id** firewall policy name or id
- **rule_name_or_id** firewall rule name or id
- **filters** (*dict*) optional filters

Raises
DuplicateResource on multiple matches

Raises
NotFoundException if firewall policy is not found

Returns
updated firewall policy

Return type
FirewallPolicy

remove_server_security_groups(*server*, *security_groups*)

Remove security groups from a server

Remove existing security groups from an existing server. If the security groups are not present on the server this will continue unaffected.

Parameters

- **server** The server to remove security groups from.
- **security_groups** A list of security groups to remove.

Returns
False if server or security groups are undefined, True otherwise.

Raises
SDKException on operation error.

remove_user_from_group(*name_or_id*, *group_name_or_id*)

Remove a user from a group.

Parameters

- **name_or_id** Name or unique ID of the user.
- **group_name_or_id** Group name or ID

Raises
SDKException if something goes wrong during the OpenStack API call

remove_volume_type_access(*name_or_id*, *project_id*)

Revoke access on a volume_type to a project.

Parameters

- **name_or_id** ID or name of a volume_type
- **project_id** A project id

Returns
None

Raises
SDKException on operation error.

revoke_role(*name_or_id*, *user=None*, *group=None*, *project=None*, *domain=None*, *system=None*, *inherited=False*, *wait=False*, *timeout=60*)

Revoke a role from a user.

Parameters

- **name_or_id** (*string*) Name or unique ID of the role.
- **user** (*string*) The name or id of the user.
- **group** (*string*) The name or id of the group. (v3)
- **project** (*string*) The name or id of the project.
- **domain** (*string*) The id of the domain. (v3)
- **system** (*bool*) The name of the system. (v3)
- **inherited** (*bool*) Whether the role assignment is inherited.
- **wait** (*bool*) Wait for role to be revoked
- **timeout** (*int*) Timeout to wait for role to be revoked

  NOTE: for wait and timeout, sometimes revoking roles is not instantaneous.

  NOTE: project is required for keystone v2

NOTE: precedence is given first to project, then domain, then system

Returns
True if the role is revoke, otherwise False

Raises
SDKException if the role cannot be removed

search_aggregates(*name_or_id=None*, *filters=None*)

Seach host aggregates.

Parameters

- **name** Aggregate name or id.
- **filters** (*dict*) A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{
    'availability_zone': 'nova',
    'metadata': {'cpu_allocation_ratio': '1.0'},
}
```

Returns
A list of compute `Aggregate` objects matching the search criteria.

Raises
SDKException if something goes wrong during the OpenStack API call.

search_cluster_templates(*name_or_id=None*, *filters=None*, *detail=False*)

Search cluster templates.

Parameters

- **name_or_id** cluster template name or ID.
- **filters** a dict containing additional filters to use.
- **detail** a boolean to control if we need summarized or detailed output.

Returns
a list of dict containing the cluster templates

Raises
SDKException: if something goes wrong during the OpenStack API call.

search_coe_clusters(*name_or_id=None*, *filters=None*)

Search COE cluster.

Parameters

- **name_or_id** cluster name or ID.
- **filters** a dict containing additional filters to use.
- **detail** a boolean to control if we need summarized or detailed output.

Returns
A list of container infrastructure management `Cluster` objects.

Raises
SDKException if something goes wrong during the OpenStack API call.

search_containers(*name=None*, *filters=None*)

Search containers.

Parameters

- **name** (*string*) Container name.
- **filters** A dict containing additional filters to use. OR A string containing a jmespath expression for further filtering. Example:: [?last_name==‘Smith‘] | [?other.gender]==‘Female‘]

Returns
A list of object store `Container` objects matching the search criteria.

Raises
SDKException: If something goes wrong during the OpenStack API call.

search_domains(*filters=None*, *name_or_id=None*)

Search Keystone domains.

Parameters

- **name_or_id** Name or ID of the domain(s).
- **filters** dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR

A string containing a jmespath expression for further filtering. Example:

```
"[?last_name==`Smith`] | [?other.gender]==`Female`]"
```

Returns

a list of identity `Domain` objects

Raises

SDKException if something goes wrong during the OpenStack API call.

search_endpoints(*id=None*, *filters=None*)

List Keystone endpoints.

Parameters

- **id** ID of endpoint(s).
- **filters** dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR

A string containing a jmespath expression for further filtering. Example:

```
"[?last_name==`Smith`] | [?other.gender]==`Female`]"
```

Returns

A list of identity `Endpoint` objects

Raises

SDKException if something goes wrong during the OpenStack API call.

search_flavors(*name_or_id=None*, *filters=None*, *get_extra=True*)

Search flavors.

Parameters

- **name_or_id** Name or unique ID of the flavor(s).
- **filters**
- **get_extra** Whether to fetch extra specs.

Returns
A list of compute `Flavor` objects matching the search criteria.

search_groups(*name_or_id=None*, *filters=None*, *domain_id=None*)

Search Keystone groups.

Parameters

- **name_or_id** Name or ID of the group(s).
- **filters** dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR

A string containing a jmespath expression for further filtering. Example:

```
"[?last_name==`Smith`] | [?other.gender]==`Female`]"
```

- **domain_id** Domain ID to scope the searched groups.

Returns
A list of identity `Group` objects

Raises
SDKException if something goes wrong during the OpenStack API call.

search_keypairs(*name_or_id=None*, *filters=None*)

Search keypairs.

Parameters

- **name_or_id** Name or unique ID of the keypair(s).
- **filters** A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR

A string containing a jmespath expression for further filtering. Invalid filters will be ignored.

Returns
A list of compute `Keypair` objects matching the search criteria.

search_networks(*name_or_id=None*, *filters=None*)

Search networks

Parameters

- **name_or_id** Name or ID of the desired network.
- **filters** A dict containing additional filters to use. e.g. {router:external: True}

Returns
A list of network `Network` objects matching the search criteria.

Raises
SDKException if something goes wrong during the OpenStack API call.

search_objects(*container*, *name=None*, *filters=None*)

Search objects.

Parameters

- **name** (*string*) Object name.
- **filters** A dict containing additional filters to use. OR A string containing a jmespath expression for further filtering. Example:: [?last_name==‘Smith‘] | [?other.gender]==‘Female‘]

Returns
A list of object store `Object` objects matching the search criteria.

Raises
SDKException: If something goes wrong during the OpenStack API call.

search_ports(*name_or_id=None*, *filters=None*)

Search ports

Parameters

- **name_or_id** Name or ID of the desired port.
- **filters** A dict containing additional filters to use. e.g. {device_id: 2711c67a-b4a7-43dd-ace7-6187b791c3f0}

Returns
A list of network `Port` objects matching the search criteria.

Raises
SDKException if something goes wrong during the OpenStack API call.

search_projects(*name_or_id=None*, *filters=None*, *domain_id=None*)

Backwards compatibility method for search_projects

search_projects originally had a parameter list that was name_or_id, filters and list had domain_id first. This method exists in this form to allow code written with positional parameter to still work. But really, use keyword arguments.

Parameters

- **name_or_id** Name or ID of the project(s).
- **filters** dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR

A string containing a jmespath expression for further filtering. Example:

```
"[?last_name==`Smith`] | [?other.gender]==`Female`]"
```

- **domain_id** Domain ID to scope the searched projects.

Returns
A list of identity `Project` objects.

search_qos_bandwidth_limit_rules(*policy_name_or_id*, *rule_id=None*, *filters=None*)
Search QoS bandwidth limit rules

Parameters

- **policy_name_or_id** (*string*) Name or ID of the QoS policy to which rules should be associated.
- **rule_id** (*string*) ID of searched rule.
- **filters** a dict containing additional filters to use. e.g. {max_kbps: 1000}

Returns
A list of network `QoSBandwidthLimitRule` objects matching the search criteria.

Raises
SDKException if something goes wrong during the OpenStack API call.

search_qos_dscp_marking_rules(*policy_name_or_id*, *rule_id=None*, *filters=None*)
Search QoS DSCP marking rules

Parameters

- **policy_name_or_id** (*string*) Name or ID of the QoS policy to which rules should be associated.
- **rule_id** (*string*) ID of searched rule.
- **filters** a dict containing additional filters to use. e.g. {dscp_mark: 32}

Returns
A list of network `QoSDSCPMarkingRule` objects matching the search criteria.

Raises
SDKException if something goes wrong during the OpenStack API call.

search_qos_minimum_bandwidth_rules(*policy_name_or_id*, *rule_id=None*, *filters=None*)
Search QoS minimum bandwidth rules

Parameters

- **policy_name_or_id** (*string*) Name or ID of the QoS policy to which rules should be associated.
- **rule_id** (*string*) ID of searched rule.
- **filters** a dict containing additional filters to use. e.g. {min_kbps: 1000}

Returns
A list of network `QoSMinimumBandwidthRule` objects matching the search criteria.

Raises

SDKException if something goes wrong during the OpenStack API call.

search_qos_policies(*name_or_id=None*, *filters=None*)

Search QoS policies

Parameters

- **name_or_id** Name or ID of the desired policy.
- **filters** a dict containing additional filters to use. e.g. {shared: True}

Returns

A list of network `QosPolicy` objects matching the search criteria.

Raises

SDKException if something goes wrong during the OpenStack API call.

search_resources(*resource_type*, *name_or_id*, *get_args=None*, *get_kwargs=None*, *list_args=None*, *list_kwargs=None*, ***filters*)

Search resources

Search resources matching certain conditions

Parameters

- **resource_type** (*str*) String representation of the expected resource as *service.resource* (i.e. network.security_group).
- **name_or_id** (*str*) Name or ID of the resource
- **get_args** (*list*) Optional args to be passed to the _get call.
- **get_kwargs** (*dict*) Optional kwargs to be passed to the _get call.
- **list_args** (*list*) Optional args to be passed to the _list call.
- **list_kwargs** (*dict*) Optional kwargs to be passed to the _list call
- **filters** (*dict*) Additional filters to be used for querying resources.

search_roles(*name_or_id=None*, *filters=None*, *domain_id=None*)

Seach Keystone roles.

Parameters

- **name** Name or ID of the role(s).
- **filters** dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR

A string containing a jmespath expression for further filtering. Example:

```
"[?last_name==`Smith`] | [?other.gender]==`Female`]"
```

- **domain_id** Domain ID to scope the searched roles.

Returns
a list of identity `Role` objects

Raises
SDKException if something goes wrong during the OpenStack API call.

search_routers(*name_or_id=None*, *filters=None*)

Search routers

Parameters

- **name_or_id** Name or ID of the desired router.
- **filters** A dict containing additional filters to use. e.g. {admin_state_up: True}

Returns
A list of network `Router` objects matching the search criteria.

Raises
SDKException if something goes wrong during the OpenStack API call.

search_server_groups(*name_or_id=None*, *filters=None*)

Search server groups.

Parameters

- **name_or_id** Name or unique ID of the server group(s).
- **filters** A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR

A string containing a jmespath expression for further filtering. Invalid filters will be ignored.

Returns
A list of compute `ServerGroup` objects matching the search criteria.

Raises
SDKException if something goes wrong during the OpenStack API call.

search_servers(*name_or_id=None*, *filters=None*, *detailed=False*, *all_projects=False*, *bare=False*)

Search servers.

Parameters

- **name_or_id** Name or unique ID of the server(s).
- **filters** A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR

A string containing a jmespath expression for further filtering. Invalid filters will be ignored.

- **detailed**
- **all_projects**
- **bare**

Returns
A list of compute `Server` objects matching the search criteria.

search_services(*name_or_id=None*, *filters=None*)

Search Keystone services.

Parameters

- **name_or_id** Name or ID of the service(s).
- **filters** dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR

A string containing a jmespath expression for further filtering. Example:

```
"[?last_name==`Smith`] | [?other.gender]==`Female`]"
```

Returns
a list of identity `Service` objects

Raises
SDKException if something goes wrong during the OpenStack API call.

search_stacks(*name_or_id=None*, *filters=None*)

Search stacks.

Parameters

- **name_or_id** Name or ID of the desired stack.
- **filters** a dict containing additional filters to use. e.g. {stack_status: CREATE_COMPLETE}

Returns
a list of `openstack.orchestration.v1.stack.Stack` containing the stack description.

Raises
SDKException if something goes wrong during the OpenStack API call.

search_subnets(*name_or_id=None*, *filters=None*)

Search subnets

Parameters

- **name_or_id** Name or ID of the desired subnet.

- **filters** A dict containing additional filters to use. e.g. {enable_dhcp: True}

Returns
A list of network `Subnet` objects matching the search criteria.

Raises
SDKException if something goes wrong during the OpenStack API call.

search_users(*name_or_id=None*, *filters=None*, *domain_id=None*)

Search users.

Parameters

- **name_or_id** Name or ID of the user(s).
- **domain_id** Domain ID to scope the retrieved users.
- **filters** dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR

A string containing a jmespath expression for further filtering. Example:

```
"[?last_name==`Smith`] | [?other.gender]==`Female`]"
```

Returns
A list of identity `User` objects

Raises
SDKException if something goes wrong during the OpenStack API call.

search_volume_backups(*name_or_id=None*, *filters=None*)

Search for one or more volume backups.

Parameters

- **name_or_id** Name or unique ID of volume backup(s).
- **filters DEPRECATED** A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR

A string containing a jmespath expression for further filtering. Example:

```
"[?last_name==`Smith`] | [?other.gender]==`Female`]"
```

Returns
A list of volume `Backup` objects, if any are found.

search_volume_snapshots(*name_or_id=None*, *filters=None*)

Search for one or more volume snapshots.

Parameters

- **name_or_id** Name or unique ID of volume snapshot(s).
- **filters DEPRECATED** A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR

A string containing a jmespath expression for further filtering. Example:

```
"[?last_name==`Smith`] | [?other.gender]==`Female`]"
```

Returns
A list of volume `Snapshot` objects, if any are found.

search_volume_types(*name_or_id=None*, *filters=None*, *get_extra=None*)

Search for one or more volume types.

Parameters

- **name_or_id** Name or unique ID of volume type(s).
- **filters DEPRECATED** A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR

A string containing a jmespath expression for further filtering. Example:

```
"[?last_name==`Smith`] | [?other.gender]==`Female`]"
```

Returns
A list of volume `Type` objects, if any are found.

search_volumes(*name_or_id=None*, *filters=None*)

Search for one or more volumes.

Parameters

- **name_or_id** Name or unique ID of volume(s).
- **filters DEPRECATED** A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR

A string containing a jmespath expression for further filtering. Example:

```
"[?last_name==`Smith`] | [?other.gender]==`Female`]"
```

Returns
A list of volume `Volume` objects, if any are found.

set_aggregate_metadata(*name_or_id*, *metadata*)

Set aggregate metadata, replacing the existing metadata.

Parameters

- **name_or_id** Name of the host aggregate to update
- **metadata** Dict containing metadata to replace (Use {key: None} to remove a key)

Returns

a dict representing the new host aggregate.

Raises

SDKException on operation error.

set_compute_quotas(*name_or_id*, ***kwargs*)

Set a quota in a project

Parameters

- **name_or_id** project name or id
- **kwargs** key/value pairs of quota name and quota value

Raises

SDKException if the resource to set the quota does not exist.

set_container_access(*name*, *access*, *refresh=False*)

Set the access control list on a container.

Parameters

- **name** (*str*) Name of the container.
- **access** (*str*) ACL string to set on the container. Can also be `public` or `private` which will be translated into appropriate ACL strings.
- **refresh** Flag to trigger refresh of the container properties

set_flavor_specs(*flavor_id*, *extra_specs*)

Add extra specs to a flavor

Parameters

- **flavor_id** (*string*) ID of the flavor to update.
- **extra_specs** (*dict*) Dictionary of key-value pairs.

Returns

None

Raises

SDKException on operation error.

Raises

BadRequestException if flavor ID is not found.

set_machine_maintenance_state(*name_or_id*, *state=True*, *reason=None*)

Set Baremetal Machine Maintenance State

Sets Baremetal maintenance state and maintenance reason.

Parameters

- **name_or_id** (*string*) The Name or UUID value representing the baremetal node.
- **state** (*boolean*) The desired state of the node. True being in maintenance where as False means the machine is not in maintenance mode. This value defaults to True if not explicitly set.
- **reason** (*string*) An optional freeform string that is supplied to the baremetal API to allow for notation as to why the node is in maintenance state.

Returns
None

Raises
SDKException on operation error.

set_machine_power_off(*name_or_id*)

De-activate baremetal machine power

This is a method that sets the node power state to off.

Params string name_or_id
A string representing the baremetal node to have power turned to an off state.

Returns
None

Raises
SDKException on operation error.

set_machine_power_on(*name_or_id*)

Activate baremetal machine power

This is a method that sets the node power state to on.

Params string name_or_id
A string representing the baremetal node to have power turned to an on state.

Returns
None

Raises
SDKException on operation error.

set_machine_power_reboot(*name_or_id*)

De-activate baremetal machine power

This is a method that sets the node power state to reboot, which in essence changes the machine power state to off, and that back to on.

Params string name_or_id
A string representing the baremetal node to have power turned to an off state.

Returns
None

Raises
SDKException on operation error.

set_network_quotas(*name_or_id*, ***kwargs*)

Set a network quota in a project

Parameters

- **name_or_id** project name or id
- **kwargs** key/value pairs of quota name and quota value

Raises

SDKException if the resource to set the quota does not exist.

set_server_metadata(*name_or_id*, *metadata*)

Set metadata in a server instance.

Parameters

- **name_or_id** (*str*) The name or ID of the server instance to update.
- **metadata** (*dict*) A dictionary with the key=value pairs to set in the server instance. It only updates the key=value pairs provided. Existing ones will remain untouched.

Returns

None

Raises

SDKException on operation error.

set_volume_bootable(*name_or_id*, *bootable=True*)

Set a volumes bootable flag.

Parameters

- **name_or_id** Name or unique ID of the volume.
- **bootable** (*bool*) Whether the volume should be bootable. (Defaults to True)

Returns

None

Raises

ResourceTimeout if wait time exceeded.

Raises

SDKException on operation error.

set_volume_quotas(*name_or_id*, ***kwargs*)

Set a volume quota in a project

Parameters

- **name_or_id** project name or id
- **kwargs** key/value pairs of quota name and quota value

Returns

None

Raises

SDKException if the resource to set the quota does not exist.

sign_coe_cluster_certificate(*cluster_id*, *csr*)

Sign client key and generate the CA certificate for a cluster

Parameters

- **cluster_id** UUID of the cluster.
- **csr** Certificate Signing Request (CSR) for authenticating client key.The CSR will be used by Magnum to generate a signed certificate that client will use to communicate with the cluster.

Returns

a dict representing the signed certs.

Raises

SDKException on operation error.

stream_object(*container*, *obj*, *query_string=None*, *resp_chunk_size=1024*)

Download the content via a streaming iterator.

Parameters

- **container** (*string*) Name of the container.
- **obj** (*string*) Name of the object.
- **query_string** (*string*) Query args for uri. (delimiter, prefix, etc.)
- **resp_chunk_size** (*int*) Chunk size of data to read. Only used if the results are

Returns

An iterator over the content or None if the object is not found.

Raises

SDKException on operation error.

unbind_accelerator_request(*uuid*, *properties*)

Unbind an accelerator from VM.

Parameters

- **uuid** The uuid of the accelerator_request to be unbinded.
- **properties** The info of VM that will unbind the accelerator.

Returns

True if unbind succeeded, False otherwise.

unregister_machine(*nics*, *uuid*, *wait=None*, *timeout=600*)

Unregister Baremetal from Ironic

Removes entries for Network Interfaces and baremetal nodes from an Ironic API

Parameters

- **nics** An array of strings that consist of MAC addresses to be removed.
- **uuid** (*string*) The UUID of the node to be deleted.
- **wait** DEPRECATED, do not use.

- **timeout** Integer value, representing seconds with a default value of 600, which controls the maximum amount of time to block until a lock is released on machine.

Raises
SDKException on operation failure.

unset_flavor_specs(*flavor_id*, *keys*)

Delete extra specs from a flavor

Parameters

- **flavor_id** (*string*) ID of the flavor to update.
- **keys** List of spec keys to delete.

Returns
None

Raises
SDKException on operation error.

Raises
BadRequestException if flavor ID is not found.

update_aggregate(*name_or_id*, ***kwargs*)

Update a host aggregate.

Parameters

- **name_or_id** Name or ID of the aggregate being updated.
- **name** New aggregate name
- **availability_zone** Availability zone to assign to hosts

Returns
The updated compute `Aggregate` object.

Raises
SDKException on operation error.

update_cluster_template(*name_or_id*, ***kwargs*)

Update a cluster template.

Parameters
name_or_id Name or ID of the cluster template being updated.

Returns
an update cluster template.

Raises
SDKException on operation error.

update_coe_cluster(*name_or_id*, ***kwargs*)

Update a COE cluster.

Parameters

- **name_or_id** Name or ID of the COE cluster being updated.
- **kwargs** Cluster attributes to be updated.

Returns
The updated cluster `Cluster` object.

Raises
SDKException on operation error.

update_container(*name*, *headers*)

Update the metadata in a container.

Parameters

- **name** (*str*) Name of the container to update.
- **headers** (*dict*) Key/Value headers to set on the container.

update_domain(*domain_id=None*, *name=None*, *description=None*, *enabled=None*, *name_or_id=None*)

Update a Keystone domain

Parameters

- **domain_id**
- **name**
- **description**
- **enabled**
- **name_or_id** Name or unique ID of the domain.

Returns
The updated identity `Domain` object.

Raises
SDKException if the domain cannot be updated

update_firewall_group(*name_or_id*, *filters=None*, ***kwargs*)

Updates firewall group. To unset egress- or ingress firewall policy, set egress_firewall_policy or ingress_firewall_policy to None. You can also set egress_firewall_policy_id and ingress_firewall_policy_id directly, which will skip the policy lookups.

Parameters

- **name_or_id** firewall group name or id
- **filters** (*dict*) optional filters
- **kwargs** firewall group update parameters See create_firewall_group docstring for valid parameters.

Returns
The updated network `FirewallGroup` object.

Raises
BadRequestException if parameters are malformed

Raises
DuplicateResource on multiple matches

Raises
NotFoundException if firewall group, a firewall policy (egress, ingress) or port is not found

update_firewall_policy(*name_or_id*, *filters=None*, ***kwargs*)

Updates firewall policy.

Parameters

- **name_or_id** firewall policy name or id
- **filters** A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR A string containing a jmespath expression for further filtering. Example:: [?last_name==‘Smith‘] | [?other.gender]==‘Female‘]

- **kwargs** firewall policy update parameters See create_firewall_policy docstring for valid parameters.

Returns
The updated network `FirewallPolicy` object.

Raises
BadRequestException if parameters are malformed

Raises
DuplicateResource on multiple matches

Raises
NotFoundException if resource is not found

update_firewall_rule(*name_or_id*, *filters=None*, ***kwargs*)

Updates firewall rule.

Parameters

- **name_or_id** firewall rule name or id
- **filters** A dictionary of meta data to use for further filtering. Elements of this dictionary may, themselves, be dictionaries. Example:

```
{'last_name': 'Smith', 'other': {'gender': 'Female'}}
```

OR A string containing a jmespath expression for further filtering. Example:: [?last_name==‘Smith‘] | [?other.gender]==‘Female‘]

- **kwargs** firewall rule update parameters. See create_firewall_rule docstring for valid parameters.

Returns
The updated network `FirewallRule` object.

Raises
BadRequestException if parameters are malformed

Raises
NotFoundException if resource is not found

update_group(*name_or_id*, *name=None*, *description=None*, ***kwargs*)

Update an existing group

Parameters

- **name_or_id** Name or unique ID of the group.
- **name** New group name.
- **description** New group description.

Returns

The updated identity `Group` object.

Raises

SDKException if something goes wrong during the OpenStack API call.

update_machine(*name_or_id*, ***attrs*)

Update a machine with new configuration information

A user-friendly method to perform updates of a machine, in whole or part.

Parameters

- **name_or_id** (*string*) A machine name or UUID to be updated.
- **attrs** Attributes to updated on the machine.

Returns

Dictionary containing a machine sub-dictonary consisting of the updated data returned from the API update operation, and a list named changes which contains all of the API paths that received updates.

Raises

SDKException on operation error.

update_network(*name_or_id*, ***kwargs*)

Update a network.

Parameters

- **name_or_id** (*string*) Name or ID of the network being updated.
- **name** (*string*) New name of the network.
- **shared** (*bool*) Set the network as shared.
- **admin_state_up** (*bool*) Set the network administrative state to up.
- **external** (*bool*) Whether this network is externally accessible.
- **provider** (*dict*) A dict of network provider options. Example:

```
{'network_type': 'vlan', 'segmentation_id': 'vlan1'}
```

- **mtu_size** (*int*) New maximum transmission unit value to address fragmentation. Minimum value is 68 for IPv4, and 1280 for IPv6.
- **port_security_enabled** (*bool*) Enable or disable port security.
- **dns_domain** (*string*) Specify the DNS domain associated with this network.

Returns
The updated network `Network` object.

Raises
SDKException on operation error.

update_object(*container*, *name*, *metadata=None*, ***headers*)

Update the metadata of an object

Parameters

- **container** The name of the container the object is in
- **name** Name for the object within the container.
- **metadata** This dict will get changed into headers that set metadata of the object
- **headers** These will be passed through to the object update API as HTTP Headers.

Returns
None

Raises
SDKException on operation error.

update_port(*name_or_id*, ***kwargs*)

Update a port

Note: to unset an attribute use None value. To leave an attribute untouched just omit it.

Parameters

- **name_or_id** name or ID of the port to update. (Required)
- **name** A symbolic name for the port. (Optional)
- **admin_state_up** The administrative status of the port, which is up (true) or down (false). (Optional)
- **fixed_ips** List of ip_addresses and subnet_ids. (Optional) If you specify only a subnet ID, OpenStack Networking allocates an available IP from that subnet to the port. If you specify both a subnet ID and an IP address, OpenStack Networking tries to allocate the specified address to the port. For example:

```
[
    {
        "ip_address": "10.29.29.13",
        "subnet_id": "a78484c4-c380-4b47-85aa-
↪21c51a2d8cbd",
    },
    ...,
]
```

- **security_groups** List of security group UUIDs. (Optional)
- **allowed_address_pairs** Allowed address pairs list (Optional) For example:

```
[
    {
        "ip_address": "23.23.23.1",
        "mac_address": "fa:16:3e:c4:cd:3f",
    },
    ...,
]
```

- **extra_dhcp_opts** Extra DHCP options. (Optional). For example:

```
[{"opt_name": "opt name1", "opt_value": "value1"}, ...]
```

- **device_owner** The ID of the entity that uses this port. For example, a DHCP agent. (Optional)
- **device_id** The ID of the resource this port is attached to.
- **vnic_type** (*binding*) The type of the created port. (Optional)
- **port_security_enabled** The security port state created on the network. (Optional)
- **qos_policy_id** The ID of the QoS policy to apply for port.

Returns
The updated network `Port` object.

Raises
SDKException on operation error.

update_project(*name_or_id*, *enabled=None*, *domain_id=None*, ***kwargs*)

Update a project

Parameters

- **name_or_id** Name or unique ID of the project.
- **enabled** Whether the project is enabled or not.
- **domain_id** Domain ID to scope the retrieved project.

Returns
An identity `Project` object.

update_qos_bandwidth_limit_rule(*policy_name_or_id*, *rule_id*, ***kwargs*)

Update a QoS bandwidth limit rule.

Parameters

- **policy_name_or_id** (*string*) Name or ID of the QoS policy to which rule is associated.
- **rule_id** (*string*) ID of rule to update.
- **max_kbps** (*int*) Maximum bandwidth limit value (in kilobits per second).
- **max_burst_kbps** (*int*) Maximum burst value (in kilobits).
- **direction** (*string*) Ingress or egress. The direction in which the traffic will be limited.

Returns
The updated network `QoSBandwidthLimitRule` object.

Raises
SDKException on operation error.

update_qos_dscp_marking_rule(*policy_name_or_id*, *rule_id*, ***kwargs*)

Update a QoS DSCP marking rule.

Parameters

- **policy_name_or_id** (*string*) Name or ID of the QoS policy to which rule is associated.
- **rule_id** (*string*) ID of rule to update.
- **dscp_mark** (*int*) DSCP mark value

Returns
The updated network `QoSDSCPMarkingRule` object.

Raises
SDKException on operation error.

update_qos_minimum_bandwidth_rule(*policy_name_or_id*, *rule_id*, ***kwargs*)

Update a QoS minimum bandwidth rule.

Parameters

- **policy_name_or_id** (*string*) Name or ID of the QoS policy to which rule is associated.
- **rule_id** (*string*) ID of rule to update.
- **min_kbps** (*int*) Minimum bandwidth value (in kilobits per second).
- **direction** (*string*) Ingress or egress. The direction in which the traffic will be available.

Returns
The updated network `QoSMinimumBandwidthRule` object.

Raises
SDKException on operation error.

update_qos_policy(*name_or_id*, ***kwargs*)

Update an existing QoS policy.

Parameters

- **name_or_id** (*string*) Name or ID of the QoS policy to update.
- **policy_name** (*string*) The new name of the QoS policy.
- **description** (*string*) The new description of the QoS policy.
- **shared** (*bool*) If True, the QoS policy will be set as shared.
- **default** (*bool*) If True, the QoS policy will be set as default for project.

Returns
The updated network `QosPolicyRule` object.

Raises
SDKException on operation error.

update_recordset(*zone*, *name_or_id*, ***kwargs*)

Update a recordset.

Parameters

- **zone** Name, ID or *openstack.dns.v2.zone.Zone* instance of the zone managing the recordset.
- **name_or_id** Name or ID of the recordset being updated.
- **records** List of the recordset definitions
- **description** Description of the recordset
- **ttl** TTL (Time to live) value in seconds of the recordset

Returns
a dict representing the updated recordset.

Raises
SDKException on operation error.

update_role(*name_or_id*, *name*, ***kwargs*)

Update a Keystone role.

Parameters

- **name_or_id** Name or unique ID of the role.
- **name** (*string*) The new role name
- **domain_id** domain id

Returns
an identity `Role` object

Raises
SDKException if the role cannot be created

update_router(*name_or_id*, *name=None*, *admin_state_up=None*, *ext_gateway_net_id=None*, *enable_snat=None*, *ext_fixed_ips=None*, *routes=None*)

Update an existing logical router.

Parameters

- **name_or_id** (*string*) The name or UUID of the router to update.
- **name** (*string*) The new router name.
- **admin_state_up** (*bool*) The administrative state of the router.
- **ext_gateway_net_id** (*string*) The network ID for the external gateway.
- **enable_snat** (*bool*) Enable Source NAT (SNAT) attribute.
- **ext_fixed_ips** List of dictionaries of desired IP and/or subnet on the external network. Example:

```
[
    {
        "subnet_id": "8ca37218-28ff-41cb-9b10-
↪039601ea7e6b",
        "ip_address": "192.168.10.2",
    }
]
```

- **routes** (*list*) A list of dictionaries with destination and nexthop parameters. To clear all routes pass an empty list ([]).

  Example:

```
[{"destination": "179.24.1.0/24", "nexthop": "172.24.3.99
↪"}]
```

Returns
The updated network `Router` object.

Raises
SDKException on operation error.

update_security_group(*name_or_id*, ***kwargs*)

Update a security group

Parameters

- **name_or_id** (*string*) Name or ID of the security group to update.
- **name** (*string*) New name for the security group.
- **description** (*string*) New description for the security group.

Returns
A `openstack.network.v2.security_group.SecurityGroup` describing the updated security group.

Raises
SDKException on operation error.

update_server(*name_or_id*, *detailed=False*, *bare=False*, ***kwargs*)

Update a server.

Parameters

- **name_or_id** Name of the server to be updated.
- **detailed** Whether or not to add detailed additional information. Defaults to False.
- **bare** Whether to skip adding any additional information to the server record. Defaults to False, meaning the addresses dict will be populated as needed from neutron. Setting to True implies detailed = False.
- **name** New name for the server
- **description** New description for the server

Returns
The updated compute `Server` object.

Raises
SDKException on operation error.

update_stack(*name_or_id*, *template_file=None*, *template_url=None*, *template_object=None*, *files=None*, *rollback=True*, *tags=None*, *wait=False*, *timeout=3600*, *environment_files=None*, ***parameters*)

Update a stack.

Parameters

- **name_or_id** (*string*) Name or ID of the stack to update.
- **template_file** (*string*) Path to the template.
- **template_url** (*string*) URL of template.
- **template_object** (*string*) URL to retrieve template object.
- **files** (*dict*) dict of additional file content to include.
- **rollback** (*boolean*) Enable rollback on update failure.
- **wait** (*boolean*) Whether to wait for the delete to finish.
- **timeout** (*int*) Stack update timeout in seconds.
- **environment_files** Paths to environment files to apply.

Other arguments will be passed as stack parameters which will take precedence over any parameters specified in the environments.

Only one of template_file, template_url, template_object should be specified.

Returns
a dict containing the stack description

Raises
SDKException if something goes wrong during the OpenStack API calls

update_subnet(*name_or_id*, *subnet_name=None*, *enable_dhcp=None*, *gateway_ip=None*, *disable_gateway_ip=None*, *allocation_pools=None*, *dns_nameservers=None*, *host_routes=None*)

Update an existing subnet.

Parameters

- **name_or_id** (*string*) Name or ID of the subnet to update.
- **subnet_name** (*string*) The new name of the subnet.
- **enable_dhcp** (*bool*) Set to `True` if DHCP is enabled and `False` if disabled.
- **gateway_ip** (*string*) The gateway IP address. When you specify both allocation_pools and gateway_ip, you must ensure that the gateway IP does not overlap with the specified allocation pools.
- **disable_gateway_ip** (*bool*) Set to `True` if gateway IP address is disabled and `False` if enabled. It is not allowed with gateway_ip. Default is `False`.

- **allocation_pools** A list of dictionaries of the start and end addresses for the allocation pools. For example:

```
[{"start": "192.168.199.2", "end": "192.168.199.254"}]
```

- **dns_nameservers** A list of DNS name servers for the subnet. For example:

```
["8.8.8.7", "8.8.8.8"]
```

- **host_routes** A list of host route dictionaries for the subnet. For example:

```
[
    {"destination": "0.0.0.0/0", "nexthop": "123.456.78.9
↪"},
    {"destination": "192.168.0.0/24", "nexthop": "192.
↪168.0.1"},
]
```

Returns
The updated network `Subnet` object.

Raises
SDKException on operation error.

update_volume(*name_or_id*, ***kwargs*)

Update a volume.

Parameters

- **name_or_id** Name or unique ID of the volume.
- **kwargs** Volume attributes to be updated.

Returns
The updated volume `Volume` object.

update_zone(*name_or_id*, ***kwargs*)

Update a zone.

Parameters

- **name_or_id** Name or ID of the zone being updated.
- **email** Email of the zone owner (only applies if zone_type is primary)
- **description** Description of the zone
- **ttl** TTL (Time to live) value in seconds
- **masters** Master nameservers (only applies if zone_type is secondary)

Returns
a dict representing the updated zone.

Raises
SDKException on operation error.

validate_machine(*name_or_id*, *for_deploy=True*)

Validate parameters of the machine.

Parameters

- **name_or_id** (*string*) The Name or UUID value representing the baremetal node.
- **for_deploy** (*bool*) If `True`, validate readiness for deployment, otherwise validate only the power management properties.

Raises

ValidationException

volume_exists(*name_or_id*)

Check if a volume exists.

Parameters

name_or_id Name or unique ID of the volume.

Returns

True if the volume exists, else False.

wait_for_baremetal_node_lock(*node*, *timeout=30*)

Wait for a baremetal node to have no lock.

DEPRECATED, use `wait_for_node_reservation` on the *baremetal* proxy.

Raises

SDKException upon client failure.

Returns

None

wait_for_server(*server*, *auto_ip=True*, *ips=None*, *ip_pool=None*, *reuse=True*, *timeout=180*, *nat_destination=None*)

Wait for a server to reach ACTIVE status.

Transitioning from Profile

Support exists for users coming from older releases of OpenStack SDK who have been using the `Profile` interface.

Transition from Profile

Note

This section describes migrating code from a previous interface of openstacksdk and can be ignored by people writing new code.

If you have code that currently uses the `Profile` object and/or an `authenticator` instance from an object based on `openstack.auth.base.BaseAuthPlugin`, that code should be updated to use the *CloudRegion* object instead.

Important

`Profile` is going away. Existing code using it should be migrated as soon as possible.

Writing Code that Works with Both

These examples should all work with both the old and new interface, with one caveat. With the old interface, the `CloudConfig` object comes from the `os-client-config` library, and in the new interface that has been moved into the SDK. In order to write code that works with both the old and new interfaces, use the following code to import the config namespace:

```
try:
    from openstack import config as occ
except ImportError:
    from os_client_config import config as occ
```

The examples will assume that the config module has been imported in that manner.

Note

Yes, there is an easier and less verbose way to do all of these. These are verbose to handle both the old and new interfaces in the same codebase.

Replacing authenticator

There is no direct replacement for `openstack.auth.base.BaseAuthPlugin`. `openstacksdk` uses the keystoneauth library for authentication and HTTP interactions. keystoneauth has auth plugins that can be used to control how authentication is done. The `auth_type` config parameter can be set to choose the correct authentication method to be used.

Replacing Profile

The right way to replace the use of `openstack.profile.Profile` depends a bit on what youre trying to accomplish. Common patterns are listed below, but in general the approach is either to pass a cloud name to the *openstack.connection.Connection* constructor, or to construct a *openstack.config.cloud_region.CloudRegion* object and pass it to the constructor.

All of the examples on this page assume that you want to support old and new interfaces simultaneously. There are easier and less verbose versions of each that are available if you can just make a clean transition.

Getting a Connection to a named cloud from clouds.yaml

If you want is to construct a *openstack.connection.Connection* based on parameters configured in a `clouds.yaml` file, or from environment variables:

```
import openstack.connection

conn = connection.from_config(cloud_name='name-of-cloud-you-want')
```

Getting a Connection from python arguments avoiding clouds.yaml

If, on the other hand, you want to construct a *openstack.connection.Connection*, but are in a context where reading config from a clouds.yaml file is undesirable, such as inside of a Service:

- create a *openstack.config.loader.OpenStackConfig* object, telling it to not load yaml files. Optionally pass an `app_name` and `app_version` which will be added to user-agent strings.

- get a *openstack.config.cloud_region.CloudRegion* object from it
- get a *openstack.connection.Connection*

```
try:
    from openstack import config as occ
except ImportError:
    from os_client_config import config as occ
from openstack import connection

loader = occ.OpenStackConfig(
    load_yaml_files=False,
    app_name='spectacular-app',
    app_version='1.0')
cloud_region = loader.get_one_cloud(
    region_name='my-awesome-region',
    auth_type='password',
    auth=dict(
        auth_url='https://auth.example.com',
        username='amazing-user',
        user_domain_name='example-domain',
        project_name='astounding-project',
        user_project_name='example-domain',
        password='super-secret-password',
    ))
conn = connection.from_config(cloud_config=cloud_region)
```

Note

app_name and app_version are completely optional, and auth_type defaults to password. They are shown here for clarity as to where they should go if they want to be set.

Getting a Connection from python arguments and optionally clouds.yaml

If you want to make a connection from python arguments and want to allow one of them to optionally be `cloud` to allow selection of a named cloud, its essentially the same as the previous example, except without `load_yaml_files=False`.

```
try:
    from openstack import config as occ
except ImportError:
    from os_client_config import config as occ
from openstack import connection

loader = occ.OpenStackConfig(
    app_name='spectacular-app',
    app_version='1.0')
cloud_region = loader.get_one_cloud(
    region_name='my-awesome-region',
    auth_type='password',
```

(continues on next page)

(continued from previous page)

```
    auth=dict(
        auth_url='https://auth.example.com',
        username='amazing-user',
        user_domain_name='example-domain',
        project_name='astounding-project',
        user_project_name='example-domain',
        password='super-secret-password',
      ))
conn = connection.from_config(cloud_config=cloud_region)
```

Parameters to get_one_cloud

The most important things to note are:

- `auth_type` specifies which kind of authentication plugin to use. It controls how authentication is done, as well as what parameters are required.
- `auth` is a dictionary containing the parameters needed by the auth plugin. The most common information it needs are user, project, domain, auth_url and password.
- The rest of the keyword arguments to `openstack.config.loader.OpenStackConfig.get_one_cloud` are either parameters needed by the keystoneauth Session object, which control how HTTP connections are made, or parameters needed by the keystoneauth Adapter object, which control how services are found in the Keystone Catalog.

For keystoneauth Adapter parameters, since there is one *openstack.connection.Connection* object but many services, per-service parameters are formed by using the official `service_type` of the service in question. For instance, to override the endpoint for the `compute` service, the parameter `compute_endpoint_override` would be used.

`region_name` in `openstack.profile.Profile` was a per-service parameter. This is no longer a valid concept. An *openstack.connection.Connection* is a connection to a region of a cloud. If you are in an extreme situation where you have one service in one region and a different service in a different region, you must use two different *openstack.connection.Connection* objects.

Note

service_type, although a parameter for keystoneauth1.adapter.Adapter, is not a valid parameter for get_one_cloud. service_type is the key by which services are referred, so saying compute_service_type=henry doesnt have any meaning.

Once you have a *Connection* instance, services are accessed through instances of *Proxy* or subclasses of it that exist as attributes on the *Connection*.

Service Proxies

The following service proxies exist on the *Connection*. The service proxies are all always present on the *Connection* object, but the combination of your `CloudRegion` and the catalog of the cloud in question control which services can be used.

Accelerator API

The Accelerator Class

The accelerator high-level interface is available through the `accelerator` member of a *Connection* object. The `accelerator` member will only be added if the service is detected.

Device Operations

class openstack.accelerator.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

devices(***query*)

Retrieve a generator of devices.

Parameters

query (*kwargs*) Optional query parameters to be sent to restrict the devices to be returned. Available parameters include:

- hostname: The hostname of the device.
- type: The type of the device.
- vendor: The vendor ID of the device.
- sort: A list of sorting keys separated by commas. Each sorting key can optionally be attached with a sorting direction modifier which can be `asc` or `desc`.
- limit: Requests a specified size of returned items from the query. Returns a number of items up to the specified limit value.
- marker: Specifies the ID of the last-seen item. Use the limit parameter to make an initial limited request and use the ID of the last-seen item from the response as the marker parameter value in a subsequent limited request.

Returns

A generator of device instances.

get_device(*uuid*, *fields=None*)

Get a single device.

Parameters

uuid The value can be the UUID of a device.

Returns

One *Device*

Raises

NotFoundException when no device matching the criteria could be found.

Deployable Operations

class openstack.accelerator.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

deployables(***query*)

Retrieve a generator of deployables.

Parameters

query (*kwargs*) Optional query parameters to be sent to restrict the deployables to be returned.

Returns

A generator of deployable instances.

get_deployable(*uuid*, *fields=None*)

Get a single deployable.

Parameters

uuid The value can be the UUID of a deployable.

Returns

One *Deployable*

Raises

NotFoundException when no deployable matching the criteria could be found.

update_deployable(*uuid*, *patch*)

Reconfig the FPGA with new bitstream.

Parameters

- **uuid** The value can be the UUID of a deployable
- **patch** The information to reconfig.

Returns

The results of FPGA reconfig.

Device Profile Operations

class openstack.accelerator.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

device_profiles(***query*)

Retrieve a generator of device profiles.

Parameters
query (*kwargs*) Optional query parameters to be sent to restrict the device profiles to be returned.

Returns
A generator of device profile instances.

`create_device_profile`(***attrs*)

Create a device_profile.

Parameters
attrs (*kwargs*) a list of device_profiles.

Returns
The list of created device profiles

`delete_device_profile`(*device_profile*, *ignore_missing=True*)

Delete a device profile

Parameters

- **device_profile** The value can be either the ID of a device profile or a *DeviceProfile* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the device profile does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent device profile.

Returns
`None`

`get_device_profile`(*uuid*, *fields=None*)

Get a single device profile.

Parameters
uuid The value can be the UUID of a device profile.

Returns
One :class: *~openstack.accelerator.v2.device_profile.DeviceProfile*

Raises
NotFoundException when no device profile matching the criteria could be found.

Accelerator Request Operations

`class openstack.accelerator.v2._proxy.Proxy`(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

`accelerator_requests`(***query*)

Retrieve a generator of accelerator requests.

Parameters
query (*kwargs*) Optional query parameters to be sent to restrict the accelerator requests to be returned.

Returns
A generator of accelerator request instances.

create_accelerator_request(***attrs*)

Create an ARQs for a single device profile.

Parameters
attrs (*kwargs*) request body.

Returns
The created accelerator request instance.

delete_accelerator_request(*accelerator_request*, *ignore_missing=True*)

Delete a device profile

Parameters

- **device_profile** The value can be either the ID of a device profile or a *DeviceProfile* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the device profile does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent accelerator request.

Returns
`None`

get_accelerator_request(*uuid*, *fields=None*)

Get a single accelerator request.

Parameters
uuid The value can be the UUID of a accelerator request.

Returns
One :class: *~openstack.accelerator.v2.accelerator_request.AcceleratorRequest*

Raises
NotFoundException when no accelerator request matching the criteria could be found.

update_accelerator_request(*uuid*, *properties*)

Bind/Unbind an accelerator to VM.

Parameters

- **uuid** The uuid of the accelerator_request to be bound/unbound.
- **properties** The info of VM that will bind/unbind the accelerator.

Returns
True if bind/unbind succeeded, False otherwise.

Baremetal API

For details on how to use baremetal, see *Using OpenStack Baremetal*

The Baremetal Class

The baremetal high-level interface is available through the `baremetal` member of a `Connection` object. The `baremetal` member will only be added if the service is detected.

Node Operations

class openstack.baremetal.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

nodes(*details=False*, ***query*)

Retrieve a generator of nodes.

Parameters

- **details** A boolean indicating whether the detailed information for every node should be returned.
- **query** (*dict*) Optional query parameters to be sent to restrict the nodes returned. Available parameters include:
  - `associated`: Only return those which are, or are not, associated with an `instance_id`.
  - `conductor_group`: Only return those in the specified `conductor_group`.
  - `driver`: Only return those with the specified `driver`.
  - `fault`: Only return those with the specified fault type.
  - `fields`: A list containing one or more fields to be returned in the response. This may lead to some performance gain because other fields of the resource are not refreshed.
  - `instance_id`: Only return the node with this specific instance UUID or an empty set if not found.
  - `is_maintenance`: Only return those with `maintenance` set to `True` or `False`.
  - `limit`: Requests at most the specified number of nodes be returned from the query.
  - `marker`: Specifies the ID of the last-seen node. Use the `limit` parameter to make an initial limited request and use the ID of the last-seen node from the response as the `marker` value in a subsequent limited request.
  - `provision_state`: Only return those nodes with the specified `provision_state`.

- `resource_class`: Only return those with the specified `resource_class`.
- `shard`: Only return nodes matching the supplied shard key.
- `sort_dir`: Sorts the response by the requested sort direction. A valid value is `asc` (ascending) or `desc` (descending). Default is `asc`. You can specify multiple pairs of sort key and sort direction query parameters. If you omit the sort direction in a pair, the API uses the natural sorting direction of the server attribute that is provided as the `sort_key`.
- `sort_key`: Sorts the response by the this attribute value. Default is `id`. You can specify multiple pairs of sort key and sort direction query pa rameters. If you omit the sort direction in a pair, the API uses the natural sorting direction of the server attribute that is provided as the `sort_key`.

Returns
A generator of *Node*

create_node(***attrs*)

Create a new node from attributes.

See *create()* for an explanation of the initial provision state.

Parameters
attrs (*dict*) Keyword arguments that will be used to create a *Node*.

Returns
The results of node creation.

Return type
Node.

find_node(*name_or_id*, *ignore_missing=True*)

Find a single node.

Parameters

- **name_or_id** (*str*) The name or ID of a node.
- **ignore_missing** (*bool*) When set to `False`, an exception of *NotFoundException* will be raised when the node does not exist. When set to *True*‘, None will be returned when attempting to find a nonexistent node.

Returns
One *Node* object or None.

get_node(*node*, *fields=None*)

Get a specific node.

Parameters

- **node** The value can be the name or ID of a node or a *Node* instance.
- **fields** Limit the resource fields to fetch.

Returns
One *Node*

Raises
NotFoundException when no node matching the name or ID could be found.

update_node(*node*, *retry_on_conflict=True*, ***attrs*)

Update a node.

Parameters

- **node** The value can be the name or ID of a node or a *Node* instance.
- **retry_on_conflict** (*bool*) Whether to retry HTTP CONFLICT error. Most of the time it can be retried, since it is caused by the node being locked. However, when setting `instance_id`, this is a normal code and should not be retried.
- **attrs** (*dict*) The attributes to update on the node represented by the `node` parameter.

Returns
The updated node.

Return type
Node

patch_node(*node*, *patch*, *reset_interfaces=None*, *retry_on_conflict=True*)

Apply a JSON patch to the node.

Parameters

- **node** The value can be the name or ID of a node or a *Node* instance.
- **patch** JSON patch to apply.
- **reset_interfaces** (*bool*) whether to reset the node hardware interfaces to their defaults. This works only when changing drivers. Added in API microversion 1.45.
- **retry_on_conflict** (*bool*) Whether to retry HTTP CONFLICT error. Most of the time it can be retried, since it is caused by the node being locked. However, when setting `instance_id`, this is a normal code and should not be retried.

  See Update Node for details.

Returns
The updated node.

Return type
Node

set_node_provision_state(*node*, *target*, *config_drive=None*, *clean_steps=None*, *rescue_password=None*, *wait=False*, *timeout=None*, *deploy_steps=None*)

Run an action modifying nodes provision state.

This call is asynchronous, it will return success as soon as the Bare Metal service acknowledges the request.

Parameters

- **node** The value can be the name or ID of a node or a *Node* instance.

- **target** Provisioning action, e.g. `active`, `provide`. See the Bare Metal service documentation for available actions.
- **config_drive** Config drive to pass to the node, only valid for `active` and ``rebuild` targets. You can use functions from `openstack.baremetal.configdrive` to build it.
- **clean_steps** Clean steps to execute, only valid for `clean` target.
- **rescue_password** Password for the rescue operation, only valid for `rescue` target.
- **wait** Whether to wait for the node to get into the expected state. The expected state is determined from a combination of the current provision state and `target`.
- **timeout** If `wait` is set to `True`, specifies how much (in seconds) to wait for the expected state to be reached. The value of `None` (the default) means no client-side timeout.
- **deploy_steps** Deploy steps to execute, only valid for `active` and `rebuild` target.

Returns
The updated *Node*

Raises
ValueError if `config_drive`, `clean_steps`, `deploy_steps` or `rescue_password` are provided with an invalid `target`.

`get_node_boot_device`(*node*)

Get node boot device

Parameters
node The value can be the name or ID of a node or a *Node* instance.

Returns
The node boot device

`set_node_boot_device`(*node*, *boot_device*, *persistent=False*)

Set node boot device

Parameters

- **node** The value can be the name or ID of a node or a *Node* instance.
- **boot_device** Boot device to assign to the node.
- **persistent** If the boot device change is maintained after node reboot

Returns
The updated *Node*

`get_node_supported_boot_devices`(*node*)

Get supported boot devices for node

Parameters
node The value can be the name or ID of a node or a *Node* instance.

Returns
The node boot device

set_node_boot_mode(*node*, *target*)

Make a request to change nodes boot mode

Parameters

- **node** The value can be the name or ID of a node or a *Node* instance.
- **target** Boot mode to set for node, one of either uefi/bios.

set_node_secure_boot(*node*, *target*)

Make a request to change nodes secure boot state

Parameters

- **node** The value can be the name or ID of a node or a *Node* instance.
- **target** Boolean indicating secure boot state to set. True/False corresponding to on/off respectively.

inject_nmi_to_node(*node*)

Inject NMI to node.

Injects a non-maskable interrupt (NMI) message to the node. This is used when response time is critical, such as during non-recoverable hardware errors. In addition, virsh inject-nmi is useful for triggering a crashdump in Windows guests.

Parameters

node The value can be the name or ID of a node or a *Node* instance.

Returns

None

wait_for_nodes_provision_state(*nodes*, *expected_state*, *timeout=None*, *abort_on_failed_state=True*, *fail=True*)

Wait for the nodes to reach the expected state.

Parameters

- **nodes** List of nodes - name, ID or *Node* instance.
- **expected_state** The expected provisioning state to reach.
- **timeout** If `wait` is set to `True`, specifies how much (in seconds) to wait for the expected state to be reached. The value of `None` (the default) means no client-side timeout.
- **abort_on_failed_state** If `True` (the default), abort waiting if any node reaches a failure state which does not match the expected one. Note that the failure state for `enroll` -> `manageable` transition is `enroll` again.
- **fail** If set to `False` this call will not raise on timeouts and provisioning failures.

Returns

If *fail* is `True` (the default), the list of *Node* instances that reached the requested state. If *fail* is `False`, a *WaitResult* named tuple.

Raises

ResourceFailure if a node reaches an error state and `abort_on_failed_state` is `True`.

Raises
ResourceTimeout on timeout.

set_node_power_state(*node*, *target*, *wait=False*, *timeout=None*)

Run an action modifying nodes power state.

This call is asynchronous, it will return success as soon as the Bare Metal service acknowledges the request.

Parameters

- **node** The value can be the name or ID of a node or a *Node* instance.
- **target** Target power state, one of *PowerAction* or a string.
- **wait** Whether to wait for the node to get into the expected state.
- **timeout** If `wait` is set to `True`, specifies how much (in seconds) to wait for the expected state to be reached. The value of `None` (the default) means no client-side timeout.

wait_for_node_power_state(*node*, *expected_state*, *timeout=None*)

Wait for the node to reach the power state.

Parameters

- **node** The value can be the name or ID of a node or a *Node* instance.
- **timeout** How much (in seconds) to wait for the target state to be reached. The value of `None` (the default) means no timeout.

Returns
The updated *Node*

wait_for_node_reservation(*node*, *timeout=None*)

Wait for a lock on the node to be released.

Bare metal nodes in ironic have a reservation lock that is used to represent that a conductor has locked the node while performing some sort of action, such as changing configuration as a result of a machine state change.

This lock can occur during power syncronization, and prevents updates to objects attached to the node, such as ports.

Note that nothing prevents a conductor from acquiring the lock again after this call returns, so it should be treated as best effort.

Returns immediately if there is no reservation on the node.

Parameters

- **node** The value can be the name or ID of a node or a *Node* instance.
- **timeout** How much (in seconds) to wait for the lock to be released. The value of `None` (the default) means no timeout.

Returns
The updated *Node*

validate_node(*node*, *required=('boot', 'deploy', 'power')*)

Validate required information on a node.

Parameters

- **node** The value can be either the name or ID of a node or a *Node* instance.
- **required** List of interfaces that are required to pass validation. The default value is the list of minimum required interfaces for provisioning.

Returns
dict mapping interface names to *ValidationResult* objects.

Raises
ValidationException if validation fails for a required interface.

set_node_maintenance(*node*, *reason=None*)

Enable maintenance mode on the node.

Parameters

- **node** The value can be either the name or ID of a node or a *Node* instance.
- **reason** Optional reason for maintenance.

Returns
This Node instance.

unset_node_maintenance(*node*)

Disable maintenance mode on the node.

Parameters
node The value can be either the name or ID of a node or a *Node* instance.

Returns
This Node instance.

delete_node(*node*, *ignore_missing=True*)

Delete a node.

Parameters

- **node** The value can be either the name or ID of a node or a *Node* instance.
- **ignore_missing** (*bool*) When set to False, an exception *NotFoundException* will be raised when the node could not be found. When set to True, no exception will be raised when attempting to delete a non-existent node.

Returns
The instance of the node which was deleted.

Return type
Node.

delete_node(*node*, *ignore_missing=True*)

Delete a node.

Parameters

- **node** The value can be either the name or ID of a node or a *Node* instance.
- **ignore_missing** (*bool*) When set to False, an exception *NotFoundException* will be raised when the node could not be found.

When set to `True`, no exception will be raised when attempting to delete a non-existent node.

Returns
The instance of the node which was deleted.

Return type
Node.

`list_node_vendor_passthru`(*node*)

Lists vendor_passthru for a node.

Parameters
node The value can be the name or ID of a node or a *Node* instance.

Returns
A list of vendor_passthru methods for the node.

`get_node_console`(*node*)

Get the console for a node.

Parameters
node The value can be the name or ID of a node or a *Node* instance.

Returns
Connection information for the console.

`enable_node_console`(*node*)

Enable the console for a node.

Parameters
node The value can be the name or ID of a node or a *Node* instance.

Returns
None

`disable_node_console`(*node*)

Disable the console for a node.

Parameters
node The value can be the name or ID of a node or a *Node* instance.

Returns
None

Node Trait Operations

`class openstack.baremetal.v1._proxy.Proxy`(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

`add_node_trait`(*node*, *trait*)

Add a trait to a node.

Parameters

- **node** The value can be the name or ID of a node or a *Node* instance.
- **trait** trait to remove from the node.

Returns
The updated node

remove_node_trait(*node*, *trait*, *ignore_missing=True*)

Remove a trait from a node.

Parameters

- **node** The value can be the name or ID of a node or a *Node* instance.
- **trait** trait to remove from the node.
- **ignore_missing** (*bool*) When set to `False`, an exception *NotFoundException* will be raised when the trait could not be found. When set to `True`, no exception will be raised when attempting to delete a non-existent trait.

Returns
The updated *Node*

set_node_traits(*node*, *traits*)

Set traits for a node.

Removes any existing traits and adds the traits passed in to this method.

Parameters

- **node** The value can be the name or ID of a node or a *Node* instance.
- **traits** list of traits to add to the node.

Returns
The updated *Node*

Port Operations

class openstack.baremetal.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

ports(*details=False*, ***query*)

Retrieve a generator of ports.

Parameters

- **details** A boolean indicating whether the detailed information for every port should be returned.
- **query** (*dict*) Optional query parameters to be sent to restrict the ports returned. Available parameters include:
  - `address`: Only return ports with the specified physical hardware address, typically a MAC address.

- `driver`: Only return those with the specified `driver`.
- `fields`: A list containing one or more fields to be returned in the response. This may lead to some performance gain because other fields of the resource are not refreshed.
- `limit`: Requests at most the specified number of ports be returned from the query.
- `marker`: Specifies the ID of the last-seen port. Use the `limit` parameter to make an initial limited request and use the ID of the last-seen port from the response as the `marker` value in a subsequent limited request.
- `node`:only return the ones associated with this specific node (name or UUID), or an empty set if not found.
- `node_id`:only return the ones associated with this specific node UUID, or an empty set if not found.
- `portgroup`: only return the ports associated with this specific Portgroup (name or UUID), or an empty set if not found. Added in API microversion 1.24.
- `sort_dir`: Sorts the response by the requested sort direction. A valid value is `asc` (ascending) or `desc` (descending). Default is `asc`. You can specify multiple pairs of sort key and sort direction query parameters. If you omit the sort direction in a pair, the API uses the natural sorting direction of the server attribute that is provided as the `sort_key`.
- `sort_key`: Sorts the response by the this attribute value. Default is `id`. You can specify multiple pairs of sort key and sort direction query parameters. If you omit the sort direction in a pair, the API uses the natural sorting direction of the server attribute that is provided as the `sort_key`.

Returns
A generator of port instances.

create_port(***attrs*)

Create a new port from attributes.

Parameters
attrs (*dict*) Keyword arguments that will be used to create a *Port*.

Returns
The results of port creation.

Return type
Port.

find_port(*name_or_id*, *ignore_missing=True*)

Find a single port.

Parameters

- **name_or_id** (*str*) The ID of a port.
- **ignore_missing** (*bool*) When set to `False`, an exception of *NotFoundException* will be raised when the port does not exist. When set to *True*', None will be returned when attempting to find a nonexistent port.

Returns
One *Port* object or None.

get_port(*port*, *fields=None*)

Get a specific port.

Parameters

- **port** The value can be the ID of a port or a *Port* instance.
- **fields** Limit the resource fields to fetch.

Returns
One *Port*

Raises
NotFoundException when no port matching the name or ID could be found.

update_port(*port*, ***attrs*)

Update a port.

Parameters

- **port** Either the ID of a port or an instance of *Port*.
- **attrs** (*dict*) The attributes to update on the port represented by the `port` parameter.

Returns
The updated port.

Return type
Port

patch_port(*port*, *patch*)

Apply a JSON patch to the port.

Parameters

- **port** The value can be the ID of a port or a *Port* instance.
- **patch** JSON patch to apply.

Returns
The updated port.

Return type
Port

delete_port(*port*, *ignore_missing=True*)

Delete a port.

Parameters

- **port** The value can be either the ID of a port or a *Port* instance.
- **ignore_missing** (*bool*) When set to `False`, an exception *NotFoundException* will be raised when the port could not be found. When set to `True`, no exception will be raised when attempting to delete a non-existent port.

Returns
The instance of the port which was deleted.

Return type
Port.

Port Group Operations

class openstack.baremetal.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

port_groups(*details=False*, ***query*)

Retrieve a generator of port groups.

Parameters

- **details** A boolean indicating whether the detailed information for every port group should be returned.
- **query** (*dict*) Optional query parameters to be sent to restrict the port groups returned. Available parameters include:
  - address: Only return portgroups with the specified physical hardware address, typically a MAC address.
  - fields: A list containing one or more fields to be returned in the response. This may lead to some performance gain because other fields of the resource are not refreshed.
  - limit: Requests at most the specified number of portgroups returned from the query.
  - marker: Specifies the ID of the last-seen portgroup. Use the limit parameter to make an initial limited request and use the ID of the last-seen portgroup from the response as the marker value in a subsequent limited request.
  - node:only return the ones associated with this specific node (name or UUID), or an empty set if not found.
  - sort_dir: Sorts the response by the requested sort direction. A valid value is asc (ascending) or desc (descending). Default is asc. You can specify multiple pairs of sort key and sort direction query parameters. If you omit the sort direction in a pair, the API uses the natural sorting direction of the server attribute that is provided as the sort_key.
  - sort_key: Sorts the response by the this attribute value. Default is id. You can specify multiple pairs of sort key and sort direction query parameters. If you omit the sort direction in a pair, the API uses the natural sorting direction of the server attribute that is provided as the sort_key.

Returns
A generator of port group instances.

create_port_group(***attrs*)

Create a new portgroup from attributes.

Parameters

attrs (*dict*) Keyword arguments that will be used to create a *PortGroup*.

Returns

The results of portgroup creation.

Return type

PortGroup.

find_port_group(*name_or_id*, *ignore_missing=True*)

Find a single port group.

Parameters

- **name_or_id** (*str*) The name or ID of a portgroup.
- **ignore_missing** (*bool*) When set to `False`, an exception of *NotFoundException* will be raised when the port group does not exist. When set to *True*', None will be returned when attempting to find a nonexistent port group.

Returns

One *PortGroup* object or None.

get_port_group(*port_group*, *fields=None*)

Get a specific port group.

Parameters

- **port_group** The value can be the name or ID of a chassis or a *PortGroup* instance.
- **fields** Limit the resource fields to fetch.

Returns

One *PortGroup*

Raises

NotFoundException when no port group matching the name or ID could be found.

update_port_group(*port_group*, ***attrs*)

Update a port group.

Parameters

- **port_group** Either the name or the ID of a port group or an instance of *PortGroup*.
- **attrs** (*dict*) The attributes to update on the port group represented by the `port_group` parameter.

Returns

The updated port group.

Return type

PortGroup

patch_port_group(*port_group*, *patch*)

Apply a JSON patch to the port_group.

Parameters

- **port_group** The value can be the ID of a port group or a *PortGroup* instance.
- **patch** JSON patch to apply.

Returns
The updated port group.

Return type
PortGroup

delete_port_group(*port_group*, *ignore_missing=True*)

Delete a port group.

Parameters

- **port_group** The value can be either the name or ID of a port group or a *PortGroup* instance.
- **ignore_missing** (*bool*) When set to `False`, an exception *NotFoundException* will be raised when the port group could not be found. When set to `True`, no exception will be raised when attempting to delete a non-existent port group.

Returns
The instance of the port group which was deleted.

Return type
PortGroup.

Driver Operations

class openstack.baremetal.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

drivers(*details=False*, ***query*)

Retrieve a generator of drivers.

Parameters

- **details** (*bool*) A boolean indicating whether the detailed information for every driver should be returned.
- **query** (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of driver instances.

get_driver(*driver*)

Get a specific driver.

Parameters

driver The value can be the name of a driver or a *Driver* instance.

Returns

One *Driver*

Raises

NotFoundException when no driver matching the name could be found.

Chassis Operations

class openstack.baremetal.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

chassis(*details=False*, ***query*)

Retrieve a generator of chassis.

Parameters

- **details** A boolean indicating whether the detailed information for every chassis should be returned.
- **query** (*dict*) Optional query parameters to be sent to restrict the chassis to be returned. Available parameters include:
  - `fields`: A list containing one or more fields to be returned in the response. This may lead to some performance gain because other fields of the resource are not refreshed.
  - `limit`: Requests at most the specified number of items be returned from the query.
  - `marker`: Specifies the ID of the last-seen chassis. Use the `limit` parameter to make an initial limited request and use the ID of the last-seen chassis from the response as the `marker` value in a subsequent limited request.
  - `sort_dir`: Sorts the response by the requested sort direction. A valid value is `asc` (ascending) or `desc` (descending). Default is `asc`. You can specify multiple pairs of sort key and sort direction query parameters. If you omit the sort direction in a pair, the API uses the natural sorting direction of the server attribute that is provided as the `sort_key`.
  - `sort_key`: Sorts the response by the this attribute value. Default is `id`. You can specify multiple pairs of sort key and sort direction query parameters. If you omit the sort direction in a pair, the API uses the natural sorting direction of the server attribute that is provided as the `sort_key`.

Returns

A generator of chassis instances.

`create_chassis`(***attrs*)

Create a new chassis from attributes.

Parameters

attrs (*dict*) Keyword arguments that will be used to create a *Chassis*.

Returns

The results of chassis creation.

Return type

Chassis.

`find_chassis`(*name_or_id*, *ignore_missing=True*)

Find a single chassis.

Parameters

- **name_or_id** (*str*) The ID of a chassis.
- **ignore_missing** (*bool*) When set to `False`, an exception of *NotFoundException* will be raised when the chassis does not exist. When set to *True*', None will be returned when attempting to find a nonexistent chassis.

Returns

One *Chassis* object or None.

`get_chassis`(*chassis*, *fields=None*)

Get a specific chassis.

Parameters

- **chassis** The value can be the ID of a chassis or a *Chassis* instance.
- **fields** Limit the resource fields to fetch.

Returns

One *Chassis*

Raises

NotFoundException when no chassis matching the name or ID could be found.

`update_chassis`(*chassis*, ***attrs*)

Update a chassis.

Parameters

- **chassis** Either the ID of a chassis, or an instance of *Chassis*.
- **attrs** (*dict*) The attributes to update on the chassis represented by the `chassis` parameter.

Returns

The updated chassis.

Return type

Chassis

patch_chassis(*chassis*, *patch*)

Apply a JSON patch to the chassis.

Parameters

- **chassis** The value can be the ID of a chassis or a *Chassis* instance.
- **patch** JSON patch to apply.

Returns
The updated chassis.

Return type
Chassis

delete_chassis(*chassis*, *ignore_missing=True*)

Delete a chassis.

Parameters

- **chassis** The value can be either the ID of a chassis or a *Chassis* instance.
- **ignore_missing** (*bool*) When set to `False`, an exception *NotFoundException* will be raised when the chassis could not be found. When set to `True`, no exception will be raised when attempting to delete a non-existent chassis.

Returns
The instance of the chassis which was deleted.

Return type
Chassis.

VIF Operations

class openstack.baremetal.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

attach_vif_to_node(*node*, *vif_id*, *retry_on_conflict=True*, ***, *port_id=None*, *port_group_id=None*)

Attach a VIF to the node.

The exact form of the VIF ID depends on the network interface used by the node. In the most common case it is a Network service port (NOT a Bare Metal port) ID. A VIF can only be attached to one node at a time.

Parameters

- **node** The value can be either the name or ID of a node or a *Node* instance.
- **vif_id** Backend-specific VIF ID.
- **retry_on_conflict** Whether to retry HTTP CONFLICT errors. This can happen when either the VIF is already used on a node or the node is locked. Since the latter happens more often, the default value is True.

- **port_id** The UUID of the port to attach the VIF to. Only one of port_id or port_group_id can be provided.
- **port_group_id** The UUID of the portgroup to attach to. Only one of port_group_id or port_id can be provided.

Returns
None

Raises
NotSupported if the server does not support the VIF API.

Raises
InvalidRequest if both port_id and port_group_id are provided.

`detach_vif_from_node`(*node*, *vif_id*, *ignore_missing=True*)

Detach a VIF from the node.

The exact form of the VIF ID depends on the network interface used by the node. In the most common case it is a Network service port (NOT a Bare Metal port) ID.

Parameters

- **node** The value can be either the name or ID of a node or a *Node* instance.
- **vif_id** (*string*) Backend-specific VIF ID.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the VIF does not exist. Otherwise, `False` is returned.

Returns
`True` if the VIF was detached, otherwise `False`.

Raises
NotSupported if the server does not support the VIF API.

`list_node_vifs`(*node*)

List IDs of VIFs attached to the node.

The exact form of the VIF ID depends on the network interface used by the node. In the most common case it is a Network service port (NOT a Bare Metal port) ID.

Parameters
node The value can be either the name or ID of a node or a *Node* instance.

Returns
List of VIF IDs as strings.

Raises
NotSupported if the server does not support the VIF API.

Allocation Operations

`class openstack.baremetal.v1._proxy.Proxy`(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

allocations(***query*)

Retrieve a generator of allocations.

Parameters

query (*dict*) Optional query parameters to be sent to restrict the allocation to be returned. Available parameters include:

- `fields`: A list containing one or more fields to be returned in the response. This may lead to some performance gain because other fields of the resource are not refreshed.
- `limit`: Requests at most the specified number of items be returned from the query.
- `marker`: Specifies the ID of the last-seen allocation. Use the `limit` parameter to make an initial limited request and use the ID of the last-seen allocation from the response as the `marker` value in a subsequent limited request.
- `sort_dir`: Sorts the response by the requested sort direction. A valid value is `asc` (ascending) or `desc` (descending). Default is `asc`. You can specify multiple pairs of sort key and sort direction query parameters. If you omit the sort direction in a pair, the API uses the natural sorting direction of the server attribute that is provided as the `sort_key`.
- `sort_key`: Sorts the response by the this attribute value. Default is `id`. You can specify multiple pairs of sort key and sort direction query parameters. If you omit the sort direction in a pair, the API uses the natural sorting direction of the server attribute that is provided as the `sort_key`.

Returns

A generator of allocation instances.

create_allocation(***attrs*)

Create a new allocation from attributes.

Parameters

attrs (*dict*) Keyword arguments that will be used to create a *Allocation*.

Returns

The results of allocation creation.

Return type

Allocation.

get_allocation(*allocation*, *fields=None*)

Get a specific allocation.

Parameters

- **allocation** The value can be the name or ID of an allocation or a *Allocation* instance.
- **fields** Limit the resource fields to fetch.

Returns

One *Allocation*

Raises
NotFoundException when no allocation matching the name or ID could be found.

`update_allocation`(*allocation*, ***attrs*)

Update an allocation.

Parameters

- **allocation** The value can be the name or ID of an allocation or a *Allocation* instance.
- **attrs** (*dict*) The attributes to update on the allocation represented by the `allocation` parameter.

Returns
The updated allocation.

Return type
Allocation

`patch_allocation`(*allocation*, *patch*)

Apply a JSON patch to the allocation.

Parameters

- **allocation** The value can be the name or ID of an allocation or a *Allocation* instance.
- **patch** JSON patch to apply.

Returns
The updated allocation.

Return type
Allocation

`delete_allocation`(*allocation*, *ignore_missing=True*)

Delete an allocation.

Parameters

- **allocation** The value can be the name or ID of an allocation or a *Allocation* instance.
- **ignore_missing** (*bool*) When set to `False`, an exception *NotFoundException* will be raised when the allocation could not be found. When set to `True`, no exception will be raised when attempting to delete a non-existent allocation.

Returns
The instance of the allocation which was deleted.

Return type
Allocation.

`wait_for_allocation`(*allocation*, *timeout=None*, *ignore_error=False*)

Wait for the allocation to become active.

Parameters

- **allocation** The value can be the name or ID of an allocation or a *Allocation* instance.
- **timeout** How much (in seconds) to wait for the allocation. The value of None (the default) means no client-side timeout.
- **ignore_error** If True, this call will raise an exception if the allocation reaches the error state. Otherwise the error state is considered successful and the call returns.

Returns
The instance of the allocation.

Return type
Allocation.

Raises
ResourceFailure if allocation fails and ignore_error is False.

Raises
ResourceTimeout on timeout.

Volume Connector Operations

class openstack.baremetal.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

volume_connectors(*details=False*, ***query*)

Retrieve a generator of volume_connector.

Parameters

- **details** A boolean indicating whether the detailed information for every volume_connector should be returned.
- **query** (*dict*) Optional query parameters to be sent to restrict the volume_connectors returned. Available parameters include:
  - fields: A list containing one or more fields to be returned in the response. This may lead to some performance gain because other fields of the resource are not refreshed.
  - limit: Requests at most the specified number of volume_connector be returned from the query.
  - marker: Specifies the ID of the last-seen volume_connector. Use the limit parameter to make an initial limited request and use the ID of the last-seen volume_connector from the response as the marker value in subsequent limited request.
  - node:only return the ones associated with this specific node (name or UUID), or an empty set if not found.

- `sort_dir`:Sorts the response by the requested sort direction. A valid value is `asc` (ascending) or `desc` (descending). Default is `asc`. You can specify multiple pairs of sort key and sort direction query parameters. If you omit the sort direction in a pair, the API uses the natural sorting direction of the server attribute that is provided as the `sort_key`.
- `sort_key`: Sorts the response by the this attribute value. Default is `id`. You can specify multiple pairs of sort key and sort direction query parameters. If you omit the sort direction in a pair, the API uses the natural sorting direction of the server attribute that is provided as the `sort_key`.

Returns
A generator of volume_connector instances.

`create_volume_connector`(***attrs*)

Create a new volume_connector from attributes.

Parameters
attrs (*dict*) Keyword arguments that will be used to create a *VolumeConnector*.

Returns
The results of volume_connector creation.

Return type
VolumeConnector.

`find_volume_connector`(*vc_id*, *ignore_missing=True*)

Find a single volume connector.

Parameters

- **vc_id** (*str*) The ID of a volume connector.
- **ignore_missing** (*bool*) When set to `False`, an exception of *NotFoundException* will be raised when the volume connector does not exist. When set to *True*‘, None will be returned when attempting to find a nonexistent volume connector.

Returns
One `VolumeConnector` object or None.

`get_volume_connector`(*volume_connector*, *fields=None*)

Get a specific volume_connector.

Parameters

- **volume_connector** The value can be the ID of a volume_connector or a *VolumeConnector* instance.
- **fields** Limit the resource fields to fetch.‘

Returns
One :class: *~openstack.baremetal.v1.volume_connector.VolumeConnector*

Raises
NotFoundException when no volume_connector matching the name or ID could be found.‘

update_volume_connector(*volume_connector*, ***attrs*)

Update a volume_connector.

Parameters

- **volume_connector** Either the ID of a volume_connector or an instance of *VolumeConnector*.
- **attrs** (*dict*) The attributes to update on the volume_connector represented by the `volume_connector` parameter.

Returns

The updated volume_connector.

Return type

VolumeConnector

patch_volume_connector(*volume_connector*, *patch*)

Apply a JSON patch to the volume_connector.

Parameters

- **volume_connector** The value can be the ID of a volume_connector or a *VolumeConnector* instance.
- **patch** JSON patch to apply.

Returns

The updated volume_connector.

Return type

delete_volume_connector(*volume_connector*, *ignore_missing=True*)

Delete an volume_connector.

Parameters

- **volume_connector** The value can be either the ID of a volume_connector.VolumeConnector or a *VolumeConnector* instance.
- **ignore_missing** (*bool*) When set to `False`, an exception *NotFoundException* will be raised when the volume_connector could not be found. When set to `True`, no exception will be raised when attempting to delete a non-existent volume_connector.

Returns

The instance of the volume_connector which was deleted.

Return type

VolumeConnector.

Volume Target Operations

class openstack.baremetal.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

`volume_targets`(*details=False*, ***query*)

Retrieve a generator of volume_target.

Parameters

- **details** A boolean indicating whether the detailed information for every volume_target should be returned.
- **query** (*dict*) Optional query parameters to be sent to restrict the volume_targets returned. Available parameters include:
  - `fields`: A list containing one or more fields to be returned in the response. This may lead to some performance gain because other fields of the resource are not refreshed.
  - `limit`: Requests at most the specified number of volume_connector be returned from the query.
  - `marker`: Specifies the ID of the last-seen volume_target. Use the `limit` parameter to make an initial limited request and use the ID of the last-seen volume_target from the response as the `marker` value in subsequent limited request.
  - `node`:only return the ones associated with this specific node (name or UUID), or an empty set if not found.
  - `sort_dir`:Sorts the response by the requested sort direction. A valid value is `asc` (ascending) or `desc` (descending). Default is `asc`. You can specify multiple pairs of sort key and sort direction query parameters. If you omit the sort direction in a pair, the API uses the natural sorting direction of the server attribute that is provided as the `sort_key`.
  - `sort_key`: Sorts the response by the this attribute value. Default is `id`. You can specify multiple pairs of sort key and sort direction query parameters. If you omit the sort direction in a pair, the API uses the natural sorting direction of the server attribute that is provided as the `sort_key`.

Returns

A generator of volume_target instances.

`create_volume_target`(***attrs*)

Create a new volume_target from attributes.

Parameters

attrs (*dict*) Keyword arguments that will be used to create a *VolumeTarget*.

Returns

The results of volume_target creation.

Return type

VolumeTarget.

`find_volume_target`(*vt_id*, *ignore_missing=True*)

Find a single volume target.

Parameters

- **vt_id** (*str*) The ID of a volume target.

- **ignore_missing** (*bool*) When set to `False`, an exception of *NotFoundException* will be raised when the volume connector does not exist. When set to *True*', None will be returned when attempting to find a nonexistent volume target.

Returns
One `VolumeTarget` object or None.

get_volume_target(*volume_target*, *fields=None*)

Get a specific volume_target.

Parameters

- **volume_target** The value can be the ID of a volume_target or a *VolumeTarget* instance.
- **fields** Limit the resource fields to fetch.'

Returns
One *VolumeTarget*

Raises
NotFoundException when no volume_target matching the name or ID could be found.'

update_volume_target(*volume_target*, ***attrs*)

Update a volume_target.

Parameters

- **volume_target** Either the ID of a volume_target or an instance of *VolumeTarget*.
- **attrs** (*dict*) The attributes to update on the volume_target represented by the `volume_target` parameter.

Returns
The updated volume_target.

Return type
VolumeTarget

patch_volume_target(*volume_target*, *patch*)

Apply a JSON patch to the volume_target.

Parameters

- **volume_target** The value can be the ID of a volume_target or a *VolumeTarget* instance.
- **patch** JSON patch to apply.

Returns
The updated volume_target.

Return type

delete_volume_target(*volume_target*, *ignore_missing=True*)

Delete an volume_target.

Parameters

- **volume_target** The value can be either the ID of a volume_target.VolumeTarget or a *VolumeTarget* instance.
- **ignore_missing** (*bool*) When set to `False`, an exception *NotFoundException* will be raised when the volume_target could not be found. When set to `True`, no exception will be raised when attempting to delete a non-existent volume_target.

Returns
The instance of the volume_target which was deleted.

Return type
VolumeTarget.

Deploy Template Operations

class openstack.baremetal.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

deploy_templates(*details=False*, ***query*)

Retrieve a generator of deploy_templates.

Parameters

- **details** A boolean indicating whether the detailed information for every deploy_templates should be returned.
- **query** (*dict*) Optional query parameters to be sent to restrict the deploy_templates to be returned.

Returns
A generator of Deploy templates instances.

create_deploy_template(***attrs*)

Create a new deploy_template from attributes.

Parameters
attrs (*dict*) Keyword arguments that will be used to create a *DeployTemplate*.

Returns
The results of deploy_template creation.

Return type
DeployTemplate.

update_deploy_template(*deploy_template*, ***attrs*)

Update a deploy_template.

Parameters

- **deploy_template** Either the ID of a deploy_template, or an instance of *DeployTemplate*.

- **attrs** (*dict*) The attributes to update on the deploy_template represented by the `deploy_template` parameter.

Returns
The updated deploy_template.

Return type
DeployTemplate

delete_deploy_template(*deploy_template*, *ignore_missing=True*)

Delete a deploy_template.

:param deploy_template:The value can be
either the ID of a deploy_template or a *DeployTemplate* instance.

Parameters
ignore_missing (*bool*) When set to `False`, an exception:class:*~openstack.exceptions.NotFoundException* will be raised when the deploy_template could not be found. When set to `True`, no exception will be raised when attempting to delete a non-existent deploy_template.

Returns
The instance of the deploy_template which was deleted.

Return type
DeployTemplate.

get_deploy_template(*deploy_template*, *fields=None*)

Get a specific deployment template.

Parameters

- **deploy_template** The value can be the name or ID of a deployment template *DeployTemplate* instance.
- **fields** Limit the resource fields to fetch.

Returns
One *DeployTemplate*

Raises
NotFoundException when no deployment template matching the name or ID could be found.

patch_deploy_template(*deploy_template*, *patch*)

Apply a JSON patch to the deploy_templates.

Parameters

- **deploy_templates** The value can be the ID of a deploy_template or a *DeployTemplate* instance.
- **patch** JSON patch to apply.

Returns
The updated deploy_template.

Return type
DeployTemplate

Utilities

Building config drives

Helpers for building configdrive compatible with the Bare Metal service.

Baremetal Introspection API

The Baremetal Introspection Proxy

The baremetal introspection high-level interface is available through the `baremetal_introspection` member of a *Connection* object. The `baremetal_introspection` member will only be added if the service is detected.

Introspection Process Operations

class openstack.baremetal_introspection.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

introspections(***query*)

Retrieve a generator of introspection records.

Parameters

query (*dict*) Optional query parameters to be sent to restrict the records to be returned. Available parameters include:

- `fields`: A list containing one or more fields to be returned in the response. This may lead to some performance gain because other fields of the resource are not refreshed.
- `limit`: Requests at most the specified number of items be returned from the query.
- `marker`: Specifies the ID of the last-seen introspection. Use the `limit` parameter to make an initial limited request and use the ID of the last-seen introspection from the response as the `marker` value in a subsequent limited request.
- `sort_dir`: Sorts the response by the requested sort direction. A valid value is `asc` (ascending) or `desc` (descending). Default is `asc`. You can specify multiple pairs of sort key and sort direction query parameters. If you omit the sort direction in a pair, the API uses the natural sorting direction of the server attribute that is provided as the `sort_key`.
- `sort_key`: Sorts the response by the this attribute value. Default is `id`. You can specify multiple pairs of sort key and sort direction query parameters. If you omit the sort direction in a pair, the API uses the natural sorting direction of the server attribute that is provided as the `sort_key`.

Returns

A generator of *Introspection* objects

start_introspection(*node*, *manage_boot=None*)

Create a new introspection from attributes.

Parameters

- **node** The value can be either the name or ID of a node or a *Node* instance.
- **manage_boot** (*bool*) Whether to manage boot parameters for the node. Defaults to the server default (which is *True*).

Returns

Introspection instance.

get_introspection(*introspection*)

Get a specific introspection.

Parameters

introspection The value can be the name or ID of an introspection (matching bare metal node name or ID) or an *Introspection* instance.

Returns

Introspection instance.

Raises

NotFoundException when no introspection matching the name or ID could be found.

get_introspection_data(*introspection*, *processed=True*)

Get introspection data.

Parameters

- **introspection** The value can be the name or ID of an introspection (matching bare metal node name or ID) or an *Introspection* instance.
- **processed** Whether to fetch the final processed data (the default) or the raw unprocessed data as received from the ramdisk.

Returns

introspection data from the most recent successful run.

Return type

dict

abort_introspection(*introspection*, *ignore_missing=True*)

Abort an introspection.

Note that the introspection is not aborted immediately, you may use *wait_for_introspection* with *ignore_error=True*.

Parameters

- **introspection** The value can be the name or ID of an introspection (matching bare metal node name or ID) or an *Introspection* instance.
- **ignore_missing** (*bool*) When set to `False`, an exception *NotFoundException* will be raised when the introspection could not be found. When set to `True`, no exception will be raised when attempting to abort a non-existent introspection.

Returns
nothing

`wait_for_introspection`(*introspection*, *timeout=None*, *ignore_error=False*)

Wait for the introspection to finish.

Parameters

- **introspection** The value can be the name or ID of an introspection (matching bare metal node name or ID) or an *Introspection* instance.
- **timeout** How much (in seconds) to wait for the introspection. The value of `None` (the default) means no client-side timeout.
- **ignore_error** If `True`, this call will raise an exception if the introspection reaches the `error` state. Otherwise the error state is considered successful and the call returns.

Returns
Introspection instance.

Raises
ResourceFailure if introspection fails and `ignore_error` is `False`.

Raises
ResourceTimeout on timeout.

Block Storage API

For details on how to use block_storage, see *Using OpenStack Block Storage*

The BlockStorage Class

The block_storage high-level interface is available through the `block_storage` member of a *Connection* object. The `block_storage` member will only be added if the service is detected.

Volume Operations

`class openstack.block_storage.v2._proxy.Proxy`(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

`get_volume`(*volume*)

Get a single volume

Parameters
volume The value can be the ID of a volume or a *Volume* instance.

Returns
One *Volume*

Raises
NotFoundException when no resource can be found.

find_volume(*name_or_id*, *ignore_missing=True*, ***, *details=True*, *all_projects=False*)

Find a single volume

Parameters

- **volume** The name or ID a volume
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the volume does not exist.
- **details** (*bool*) When set to `False` no extended attributes will be returned. The default, `True`, will cause an object with additional attributes to be returned.
- **all_projects** (*bool*) When set to `True`, search for volume by name across all projects. Note that this will likely result in a higher chance of duplicates. Admin-only by default.

Returns

One *Volume* or None.

Raises

NotFoundException when no resource can be found.

Raises

DuplicateResource when multiple resources are found.

volumes(***, *details=True*, *all_projects=False*, ***query*)

Retrieve a generator of volumes

Parameters

- **details** (*bool*) When set to `False` no extended attributes will be returned. The default, `True`, will cause objects with additional attributes to be returned.
- **all_projects** (*bool*) When set to `True`, list volumes from all projects. Admin-only by default.
- **query** (*kwargs*) Optional query parameters to be sent to limit the volumes being returned. Available parameters include:
  - name: Name of the volume as a string.
  - status: Value of the status of the volume so that you can filter on available for example.

Returns

A generator of volume objects.

create_volume(***attrs*)

Create a new volume from attributes

Parameters

attrs (*dict*) Keyword arguments which will be used to create a *Volume*, comprised of the properties on the Volume class.

Returns

The results of volume creation

Return type
Volume

delete_volume(*volume*, *ignore_missing=True*, *force=False*)

Delete a volume

Parameters

- **volume** The value can be either the ID of a volume or a *Volume* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the volume does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent volume.
- **force** (*bool*) Whether to try forcing volume deletion.

Returns
None

extend_volume(*volume*, *size*)

Extend a volume

Parameters

- **volume** The value can be either the ID of a volume or a *Volume* instance.
- **size** New volume size

Returns
None

retype_volume(*volume*, *new_type*, *migration_policy='never'*)

Retype the volume.

Parameters

- **volume** The value can be either the ID of a volume or a *Volume* instance.
- **new_type** The new volume type that volume is changed with. The value can be either the ID of the volume type or a *Type* instance.
- **migration_policy** (*str*) Specify if the volume should be migrated when it is re-typed. Possible values are on-demand or never. Default: never.

Returns
None

set_volume_bootable_status(*volume*, *bootable*)

Set bootable status of the volume.

Parameters

- **volume** The value can be either the ID of a volume or a *Volume* instance.
- **bootable** (*bool*) Specifies whether the volume should be bootable or not.

Returns
None

set_volume_image_metadata(*volume*, ***metadata*)

Update image metadata for a volume

Parameters

- **volume** Either the ID of a volume or a *Volume*.
- **metadata** (*kwargs*) Key/value pairs to be updated in the volumes image metadata. No other metadata is modified by this call.

Returns
None

delete_volume_image_metadata(*volume*, *keys=None*)

Delete metadata for a volume

Parameters

- **volume** Either the ID of a volume or a *Volume*.
- **keys** (*list*) The keys to delete. If left empty complete metadata will be removed.

Returns
None

reset_volume_status(*volume*, *status=None*, *attach_status=None*, *migration_status=None*)

Reset volume statuses.

Parameters

- **volume** The value can be either the ID of a volume or a *Volume* instance.
- **status** (*str*) The new volume status.
- **attach_status** (*str*) The new volume attach status.
- **migration_status** (*str*) The new volume migration status (admin only).

Returns
None

attach_volume(*volume*, *mountpoint*, *instance=None*, *host_name=None*)

Attaches a volume to a server.

Parameters

- **volume** The value can be either the ID of a volume or a *Volume* instance.
- **mountpoint** (*str*) The attaching mount point.
- **instance** (*str*) The UUID of the attaching instance.
- **host_name** (*str*) The name of the attaching host.

Returns
None

detach_volume(*volume*, *attachment*, *force=False*, *connector=None*)

Detaches a volume from a server.

Parameters

- **volume** The value can be either the ID of a volume or a *Volume* instance.
- **attachment** (*str*) The ID of the attachment.

- **force** (*bool*) Whether to force volume detach (Rolls back an unsuccessful detach operation after you disconnect the volume.)
- **connector** (*dict*) The connector object.

Returns
None

unmanage_volume(*volume*)

Removes a volume from Block Storage management without removing the back-end storage object that is associated with it.

Parameters
volume The value can be either the ID of a volume or a *Volume* instance.

Returns
None

migrate_volume(*volume*, *host=None*, *force_host_copy=False*, *lock_volume=False*)

Migrates a volume to the specified host.

Parameters

- **volume** The value can be either the ID of a volume or a *Volume* instance.
- **host** (*str*) The target host for the volume migration. Host format is host@backend.
- **force_host_copy** (*bool*) If false (the default), rely on the volume backend driver to perform the migration, which might be optimized. If true, or the volume driver fails to migrate the volume itself, a generic host-based migration is performed.
- **lock_volume** (*bool*) If true, migrating an available volume will change its status to maintenance preventing other operations from being performed on the volume such as attach, detach, retype, etc.

Returns
None

complete_volume_migration(*volume*, *new_volume*, *error=False*)

Complete the migration of a volume.

Parameters

- **volume** The value can be either the ID of a volume or a *Volume* instance.
- **new_volume** (*str*) The UUID of the new volume.
- **error** (*bool*) Used to indicate if an error has occured elsewhere that requires clean up.

Returns
None

get_volume_metadata(*volume*)

Return a dictionary of metadata for a volume

Parameters
volume Either the ID of a volume or a *Volume*.

Returns
A *Volume* with the volumes metadata. All keys and values are Unicode text.

Return type
Volume

set_volume_metadata(*volume*, ***metadata*)

Update metadata for a volume

Parameters

- **volume** Either the ID of a volume or a *Volume*.
- **metadata** (*kwargs*) Key/value pairs to be updated in the volumes metadata. No other metadata is modified by this call. All keys and values are stored as Unicode.

Returns
A *Volume* with the volumes metadata. All keys and values are Unicode text.

Return type
Volume

delete_volume_metadata(*volume*, *keys=None*)

Delete metadata for a volume

Parameters

- **volume** Either the ID of a volume or a *Volume*.
- **keys** (*list*) The keys to delete. If left empty complete metadata will be removed.

Return type
None

Backup Operations

class openstack.block_storage.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

backups(*details=True*, ***query*)

Retrieve a generator of backups

Parameters

- **details** (*bool*) When set to False no additional details will be returned. The default, True, will cause objects with additional attributes to be returned.
- **query** (*dict*) Optional query parameters to be sent to limit the resources being returned:
  - offset: pagination marker

- limit: pagination limit
- sort_key: Sorts by an attribute. A valid value is name, status, container_format, disk_format, size, id, created_at, or updated_at. Default is created_at. The API uses the natural sorting direction of the sort_key attribute value.
- sort_dir: Sorts by one or more sets of attribute and sort direction combinations. If you omit the sort direction in a set, default is desc.

Returns
A generator of backup objects.

`get_backup`(*backup*)

Get a backup

Parameters
backup The value can be the ID of a backup or a *Backup* instance.

Returns
Backup instance

Return type
Backup

`create_backup`(***attrs*)

Create a new Backup from attributes with native API

Parameters
attrs (*dict*) Keyword arguments which will be used to create a *Backup* comprised of the properties on the Backup class.

Returns
The results of Backup creation

Return type
Backup

`delete_backup`(*backup*, *ignore_missing=True*, *force=False*)

Delete a CloudBackup

Parameters

- **backup** The value can be the ID of a backup or a *Backup* instance
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the zone does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent zone.
- **force** (*bool*) Whether to try forcing backup deletion

Returns
`None`

`restore_backup`(*backup*, *volume_id*, *name*)

Restore a Backup to volume

Parameters

- **backup** The value can be the ID of a backup or a *Backup* instance

- **volume_id** The ID of the volume to restore the backup to.
- **name** The name for new volume creation to restore.

Returns
Updated backup instance

Return type
Backup

Capabilities Operations

class openstack.block_storage.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

get_capabilities(*host*)

Get a backends capabilites

Parameters
host Specified backend to obtain volume stats and properties.

Returns
One :class: *~openstack.block_storage.v2.capabilites.Capabilities* instance.

Raises
NotFoundException when no resource can be found.

Limits Operations

class openstack.block_storage.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

get_limits(*project=None*)

Retrieves limits

Parameters
project A project to get limits for. The value can be either the ID of a project or an `Project` instance.

Returns
A Limits object, including both *AbsoluteLimit* and *RateLimit*

Return type
Limits

Type Operations

class openstack.block_storage.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

get_type(*type*)

Get a single type

Parameters

type The value can be the ID of a type or a *Type* instance.

Returns

One *Type*

Raises

NotFoundException when no resource can be found.

types(***query*)

Retrieve a generator of volume types

Returns

A generator of volume type objects.

create_type(***attrs*)

Create a new type from attributes

Parameters

attrs (*dict*) Keyword arguments which will be used to create a *Type*, comprised of the properties on the Type class.

Returns

The results of type creation

Return type

Type

delete_type(*type*, *ignore_missing=True*)

Delete a type

Parameters

- **type** The value can be either the ID of a type or a *Type* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the type does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent type.

Returns

None

Snapshot Operations

class openstack.block_storage.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

get_snapshot(*snapshot*)

Get a single snapshot

Parameters

snapshot The value can be the ID of a snapshot or a *Snapshot* instance.

Returns

One *Snapshot*

Raises

NotFoundException when no resource can be found.

snapshots(***, *details=True*, *all_projects=False*, ***query*)

Retrieve a generator of snapshots

Parameters

- **details** (*bool*) When set to `False` *Snapshot* objects will be returned. The default, `True`, will cause *SnapshotDetail* objects to be returned.
- **all_projects** (*bool*) When set to `True`, list snapshots from all projects. Admin-only by default.
- **query** (*kwargs*) Optional query parameters to be sent to limit the snapshots being returned. Available parameters include:
  - name: Name of the snapshot as a string.
  - volume_id: volume id of a snapshot.
  - status: Value of the status of the snapshot so that you can filter on available for example.

Returns

A generator of snapshot objects.

create_snapshot(***attrs*)

Create a new snapshot from attributes

Parameters

attrs (*dict*) Keyword arguments which will be used to create a *Snapshot*, comprised of the properties on the Snapshot class.

Returns

The results of snapshot creation

Return type

Snapshot

delete_snapshot(*snapshot*, *ignore_missing=True*)

Delete a snapshot

Parameters

- **snapshot** The value can be either the ID of a snapshot or a *Snapshot* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the snapshot does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent snapshot.

Returns

`None`

Stats Operations

class openstack.block_storage.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

backend_pools(***query*)

Returns a generator of cinder Back-end storage pools

Parameters

query (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

:returns A generator of cinder Back-end storage pools objects

QuotaClassSet Operations

class openstack.block_storage.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

get_quota_class_set(*quota_class_set='default'*)

Get a single quota class set

Only one quota class is permitted, `default`.

Parameters

quota_class_set The value can be the ID of a quota class set (only `default` is supported) or a `QuotaClassSet` instance.

Returns

One `QuotaClassSet`

Raises

NotFoundException when no resource can be found.

update_quota_class_set(*quota_class_set*, ***attrs*)

Update a QuotaClassSet.

Only one quota class is permitted, `default`.

Parameters

- **quota_class_set** Either the ID of a quota class set (only `default` is supported) or a `QuotaClassSet` instance.
- **attrs** The attributes to update on the QuotaClassSet represented by `quota_class_set`.

Returns
The updated QuotaSet

Return type
QuotaSet

QuotaSet Operations

class openstack.block_storage.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

get_quota_set(*project*, *usage=False*, ***query*)

Show QuotaSet information for the project

Parameters

- **project** ID or instance of `Project` of the project for which the quota should be retrieved
- **usage** (*bool*) When set to `True` quota usage and reservations would be filled.
- **query** (*dict*) Additional query parameters to use.

Returns
One *QuotaSet*

Raises
NotFoundException when no resource can be found.

get_quota_set_defaults(*project*)

Show QuotaSet defaults for the project

Parameters
project ID or instance of `Project` of the project for which the quota should be retrieved

Returns
One *QuotaSet*

Raises
NotFoundException when no resource can be found.

revert_quota_set(*project*, ***query*)

Reset Quota for the project/user.

Parameters

- **project** ID or instance of `Project` of the project for which the quota should be resetted.
- **query** (*dict*) Additional parameters to be used.

Returns
`None`

update_quota_set(*project*, ***attrs*)

Update a QuotaSet.

Parameters

- **project** ID or instance of `Project` of the project for which the quota should be reset.
- **attrs** The attributes to update on the QuotaSet represented by `quota_set`.

Returns
The updated QuotaSet

Return type
QuotaSet

Helpers

class openstack.block_storage.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

wait_for_status(*res*, *status='available'*, *failures=None*, *interval=2*, *wait=None*, *attribute='status'*, *callback=None*)

Wait for the resource to be in a particular status.

Parameters

- **session** The session to use for making this request.
- **resource** The resource to wait on to reach the status. The resource must have a status attribute specified via `attribute`.
- **status** Desired status of the resource.
- **failures** Statuses that would indicate the transition failed such as ERROR. Defaults to [ERROR].
- **interval** Number of seconds to wait between checks.
- **wait** Maximum number of seconds to wait for transition. Set to `None` to wait forever.
- **attribute** Name of the resource attribute that contains the status.

- **callback** A callback function. This will be called with a single value, progress. This is API specific but is generally a percentage value from 0-100.

Returns
The updated resource.

Raises
ResourceTimeout if the transition to status failed to occur in `wait` seconds.

Raises
ResourceFailure if the resource transitioned to one of the states in `failures`.

Raises
`AttributeError` if the resource does not have a `status` attribute

wait_for_delete(*res*, *interval=2*, *wait=120*, *callback=None*)

Wait for a resource to be deleted.

Parameters

- **res** The resource to wait on to be deleted.
- **interval** Number of seconds to wait before to consecutive checks.
- **wait** Maximum number of seconds to wait before the change.
- **callback** A callback function. This will be called with a single value, progress, which is a percentage value from 0-100.

Returns
The resource is returned on success.

Raises
ResourceTimeout if transition to delete failed to occur in the specified seconds.

Block Storage API

For details on how to use block_storage, see *Using OpenStack Block Storage*

The BlockStorage Class

The block_storage high-level interface is available through the `block_storage` member of a *Connection* object. The `block_storage` member will only be added if the service is detected.

Volume Operations

class openstack.block_storage.v3._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

get_volume(*volume*)

Get a single volume

Parameters

volume The value can be the ID of a volume or a *Volume* instance.

Returns

One *Volume*

Raises

NotFoundException when no resource can be found.

find_volume(*name_or_id*, *ignore_missing=True*, ***, *details=True*, *all_projects=False*)

Find a single volume

Parameters

- **snapshot** The name or ID a volume
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the volume does not exist.
- **details** (*bool*) When set to `False` no extended attributes will be returned. The default, `True`, will cause objects with additional attributes to be returned.
- **all_projects** (*bool*) When set to `True`, search for volume by name across all projects. Note that this will likely result in a higher chance of duplicates. Admin-only by default.

Returns

One *Volume*

Raises

NotFoundException when no resource can be found.

Raises

DuplicateResource when multiple resources are found.

volumes(***, *details=True*, *all_projects=False*, ***query*)

Retrieve a generator of volumes

Parameters

- **details** (*bool*) When set to `False` no extended attributes will be returned. The default, `True`, will cause objects with additional attributes to be returned.
- **all_projects** (*bool*) When set to `True`, list volumes from all projects. Admin-only by default.
- **query** (*kwargs*) Optional query parameters to be sent to limit the volumes being returned. Available parameters include:
  - name: Name of the volume as a string.
  - status: Value of the status of the volume so that you can filter on available for example.

Returns

A generator of volume objects.

`create_volume`(***attrs*)

Create a new volume from attributes

Parameters
attrs (*dict*) Keyword arguments which will be used to create a *Volume*, comprised of the properties on the Volume class.

Returns
The results of volume creation

Return type
Volume

`delete_volume`(*volume*, *ignore_missing=True*, *force=False*)

Delete a volume

Parameters

- **volume** The value can be either the ID of a volume or a *Volume* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the volume does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent volume.
- **force** (*bool*) Whether to try forcing volume deletion.

Returns
`None`

`update_volume`(*volume*, ***attrs*)

Update a volume

Parameters

- **volume** Either the ID of a volume or a *Volume* instance.
- **attrs** (*dict*) The attributes to update on the volume.

Returns
The updated volume

Return type
Volume

`get_volume_metadata`(*volume*)

Return a dictionary of metadata for a volume

Parameters
volume Either the ID of a volume or a *Volume*.

Returns
A *Volume* with the volumes metadata. All keys and values are Unicode text.

Return type
Volume

`set_volume_metadata`(*volume*, ***metadata*)

Update metadata for a volume

Parameters

- **volume** Either the ID of a volume or a *Volume*.
- **metadata** (*kwargs*) Key/value pairs to be updated in the volumes metadata. No other metadata is modified by this call. All keys and values are stored as Unicode.

Returns
A *Volume* with the volumes metadata. All keys and values are Unicode text.

Return type
Volume

delete_volume_metadata(*volume*, *keys=None*)

Delete metadata for a volume

Parameters

- **volume** Either the ID of a volume or a *Volume*.
- **keys** (*list*) The keys to delete. If left empty complete metadata will be removed.

Return type
None

extend_volume(*volume*, *size*)

Extend a volume

Parameters

- **volume** The value can be either the ID of a volume or a *Volume* instance.
- **size** New volume size

Returns
None

set_volume_readonly(*volume*, *readonly=True*)

Set a volumes read-only flag.

Parameters

- **volume** The value can be either the ID of a volume or a *Volume* instance.
- **readonly** (*bool*) Whether the volume should be a read-only volume or not.

Returns
None

retype_volume(*volume*, *new_type*, *migration_policy='never'*)

Retype the volume.

Parameters

- **volume** The value can be either the ID of a volume or a *Volume* instance.
- **new_type** The new volume type that volume is changed with. The value can be either the ID of the volume type or a *Type* instance.
- **migration_policy** (*str*) Specify if the volume should be migrated when it is re-typed. Possible values are on-demand or never. Default: never.

Returns
None

`set_volume_bootable_status`(*volume*, *bootable*)

Set bootable status of the volume.

Parameters

- **volume** The value can be either the ID of a volume or a *Volume* instance.
- **bootable** (*bool*) Specifies whether the volume should be bootable or not.

Returns
None

`set_volume_image_metadata`(*volume*, ***metadata*)

Update image metadata for a volume

Parameters

- **volume** Either the ID of a volume or a *Volume*.
- **metadata** (*kwargs*) Key/value pairs to be updated in the volumes image metadata. No other metadata is modified by this call.

Returns
None

`delete_volume_image_metadata`(*volume*, *keys=None*)

Delete metadata for a volume

Parameters

- **volume** Either the ID of a volume or a *Volume*.
- **keys** (*list*) The keys to delete. If left empty complete metadata will be removed.

Returns
None

`reset_volume_status`(*volume*, *status=None*, *attach_status=None*, *migration_status=None*)

Reset volume statuses.

Parameters

- **volume** The value can be either the ID of a volume or a *Volume* instance.
- **status** (*str*) The new volume status.
- **attach_status** (*str*) The new volume attach status.
- **migration_status** (*str*) The new volume migration status (admin only).

Returns
None

`revert_volume_to_snapshot`(*volume*, *snapshot*)

Revert a volume to its latest snapshot.

This method only support reverting a detached volume, and the volume status must be available.

Parameters

- **volume** The value can be either the ID of a volume or a *Volume* instance.
- **snapshot** The value can be either the ID of a snapshot or a *Snapshot* instance.

Returns
None

attach_volume(*volume*, *mountpoint*, *instance=None*, *host_name=None*)

Attaches a volume to a server.

Parameters

- **volume** The value can be either the ID of a volume or a *Volume* instance.
- **mountpoint** (*str*) The attaching mount point.
- **instance** (*str*) The UUID of the attaching instance.
- **host_name** (*str*) The name of the attaching host.

Returns
None

detach_volume(*volume*, *attachment*, *force=False*, *connector=None*)

Detaches a volume from a server.

Parameters

- **volume** The value can be either the ID of a volume or a *Volume* instance.
- **attachment** (*str*) The ID of the attachment.
- **force** (*bool*) Whether to force volume detach (Rolls back an unsuccessful detach operation after you disconnect the volume.)
- **connector** (*dict*) The connector object.

Returns
None

manage_volume(***attrs*)

Creates a volume by using existing storage rather than
allocating new storage.

Parameters
attrs (*dict*) Keyword arguments which will be used to create a *Volume*, comprised of the properties on the Volume class.

Returns
The results of volume creation

Return type
Volume

unmanage_volume(*volume*)

Removes a volume from Block Storage management without removing the
back-end storage object that is associated with it.

Parameters
volume The value can be either the ID of a volume or a *Volume* instance.

Returns
None

migrate_volume(*volume*, *host=None*, *force_host_copy=False*, *lock_volume=False*, *cluster=None*)

Migrates a volume to the specified host.

Parameters

- **volume** The value can be either the ID of a volume or a *Volume* instance.
- **host** (*str*) The target host for the volume migration. Host format is host@backend.
- **force_host_copy** (*bool*) If false (the default), rely on the volume backend driver to perform the migration, which might be optimized. If true, or the volume driver fails to migrate the volume itself, a generic host-based migration is performed.
- **lock_volume** (*bool*) If true, migrating an available volume will change its status to maintenance preventing other operations from being performed on the volume such as attach, detach, retype, etc.
- **cluster** (*str*) The target cluster for the volume migration. Cluster format is cluster@backend. Starting with microversion 3.16, either cluster or host must be specified. If host is specified and is part of a cluster, the cluster is used as the target for the migration.

Returns
None

complete_volume_migration(*volume*, *new_volume*, *error=False*)

Complete the migration of a volume.

Parameters

- **volume** The value can be either the ID of a volume or a *Volume* instance.
- **new_volume** (*str*) The UUID of the new volume.
- **error** (*bool*) Used to indicate if an error has occured elsewhere that requires clean up.

Returns
None

upload_volume_to_image(*volume*, *image_name*, *force=False*, *disk_format=None*, *container_format=None*, *visibility=None*, *protected=None*)

Uploads the specified volume to image service.

Parameters

- **volume** The value can be either the ID of a volume or a *Volume* instance.
- **name** (*str image*) The name for the new image.
- **force** (*bool*) Enables or disables upload of a volume that is attached to an instance.

- **disk_format** (*str*) Disk format for the new image.
- **container_format** (*str*) Container format for the new image.
- **visibility** (*str*) The visibility property of the new image.
- **protected** (*str*) Whether the new image is protected.

Returns
dictionary describing the image.

reserve_volume(*volume*)

Mark volume as reserved.

Parameters
volume The value can be either the ID of a volume or a *Volume* instance.

Returns
None

unreserve_volume(*volume*)

Unmark volume as reserved.

Parameters
volume The value can be either the ID of a volume or a *Volume* instance.

Returns
None

begin_volume_detaching(*volume*)

Update volume status to detaching.

Parameters
volume The value can be either the ID of a volume or a *Volume* instance.

Returns
None

abort_volume_detaching(*volume*)

Update volume status to in-use.

Parameters
volume The value can be either the ID of a volume or a *Volume* instance.

Returns
None

init_volume_attachment(*volume*, *connector*)

Initialize volume attachment.

Parameters

- **volume** The value can be either the ID of a volume or a *Volume* instance.
- **connector** (*dict*) The connector object.

Returns
Dictionary containing the modified connector object

terminate_volume_attachment(*volume*, *connector*)

Update volume status to in-use.

Parameters

- **volume** The value can be either the ID of a volume or a *Volume* instance.
- **connector** (*dict*) The connector object.

Returns
None

Backend Pools Operations

class openstack.block_storage.v3._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

backend_pools(***query*)

Returns a generator of cinder Back-end storage pools

Parameters
query (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

:returns A generator of cinder Back-end storage pools objects

Backup Operations

class openstack.block_storage.v3._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

backups(***, *details=True*, ***query*)

Retrieve a generator of backups

Parameters

- **details** (*bool*) When set to `False` no additional details will be returned. The default, `True`, will cause objects with additional attributes to be returned.
- **query** (*dict*) Optional query parameters to be sent to limit the resources being returned:
  - offset: pagination marker
  - limit: pagination limit
  - sort_key: Sorts by an attribute. A valid value is name, status, container_format, disk_format, size, id, created_at, or updated_at. Default

is created_at. The API uses the natural sorting direction of the sort_key attribute value.

- sort_dir: Sorts by one or more sets of attribute and sort direction combinations. If you omit the sort direction in a set, default is desc.
- project_id: Project ID to query backups for.

Returns
A generator of backup objects.

`get_backup`(*backup*)

Get a backup

Parameters
backup The value can be the ID of a backup or a *Backup* instance.

Returns
Backup instance

Return type
Backup

`find_backup`(*name_or_id*, *ignore_missing=True*, ***, *details=True*)

Find a single backup

Parameters

- **snapshot** The name or ID a backup
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the backup does not exist.
- **details** (*bool*) When set to `False` no additional details will be returned. The default, `True`, will cause objects with additional attributes to be returned.

Returns
One *Backup*

Raises
NotFoundException when no resource can be found.

Raises
DuplicateResource when multiple resources are found.

`create_backup`(***attrs*)

Create a new Backup from attributes with native API

Parameters
attrs (*dict*) Keyword arguments which will be used to create a *Backup* comprised of the properties on the Backup class.

Returns
The results of Backup creation

Return type
Backup

delete_backup(*backup*, *ignore_missing=True*, *force=False*)

Delete a CloudBackup

Parameters

- **backup** The value can be the ID of a backup or a *Backup* instance
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the zone does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent zone.
- **force** (*bool*) Whether to try forcing backup deletion

Returns

`None`

restore_backup(*backup*, *volume_id=None*, *name=None*)

Restore a Backup to volume

Parameters

- **backup** The value can be the ID of a backup or a *Backup* instance
- **volume_id** The ID of the volume to restore the backup to.
- **name** The name for new volume creation to restore.

Returns

Updated backup instance

Return type

Backup

reset_backup(*backup*, *status*)

Reset status of the backup

Parameters

- **backup** The value can be either the ID of a backup or a *Backup* instance.
- **status** (*str*) New backup status

Returns

None

Availability Zone Operations

class openstack.block_storage.v3._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

availability_zones()

Return a generator of availability zones

Returns

A generator of availability zone

Return type
AvailabilityZone

Limits Operations

class openstack.block_storage.v3._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

get_limits(*project=None*)

Retrieves limits

Parameters
project A project to get limits for. The value can be either the ID of a project or an *Project* instance.

Returns
A Limits object, including both *AbsoluteLimit* and *RateLimit*

Return type
Limits

Capabilities Operations

class openstack.block_storage.v3._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

get_capabilities(*host*)

Get a backends capabilites

Parameters
host Specified backend to obtain volume stats and properties.

Returns
One :class: *~openstack.block_storage.v3.capabilites.Capabilities* instance.

Raises
NotFoundException when no resource can be found.

Group Operations

class openstack.block_storage.v3._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

get_group(*group_id*, ***attrs*)

Get a group

Parameters

- **group_id** The ID of the group to get.
- **attrs** (*dict*) Optional query parameters to be sent to limit the resources being returned.

Returns

A Group instance.

Return type

group

find_group(*name_or_id*, *ignore_missing=True*, ***, *details=True*)

Find a single group

Parameters

- **name_or_id** The name or ID of a group.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the group snapshot does not exist.
- **details** (*bool*) When set to `False`, no additional details will be returned. The default, `True`, will cause additional details to be returned.

Returns

One *Group*

Raises

NotFoundException when no resource can be found.

Raises

DuplicateResource when multiple resources are found.

groups(***, *details=True*, ***query*)

Retrieve a generator of groups

Parameters

- **details** (*bool*) When set to `False`, no additional details will be returned. The default, `True`, will cause additional details to be returned.
- **query** (*dict*) Optional query parameters to be sent to limit the resources being returned:
  - all_tenants: Shows details for all project.
  - sort: Comma-separated list of sort keys and optional sort directions.
  - limit: Returns a number of items up to the limit value.
  - offset: Used in conjunction with limit to return a slice of items. Specifies where to start in the list.
  - marker: The ID of the last-seen item.
  - list_volume: Show volume ids in this group.
  - detailed: If True, will list groups with details.

– search_opts: Search options.

Returns
A generator of group objects.

`create_group`(***attrs*)

Create a new group from attributes

Parameters
attrs (*dict*) Keyword arguments which will be used to create a *Group* comprised of the properties on the Group class.

Returns
The results of group creation.

Return type
Group.

`create_group_from_source`(***attrs*)

Creates a new group from source

Parameters
attrs (*dict*) Keyword arguments which will be used to create a *Group* comprised of the properties on the Group class.

Returns
The results of group creation.

Return type
Group.

`reset_group_state`(*group*, *status*)

Reset group status

Parameters

- **group** The *Group* to set the state.
- **status** The status for a group.

Returns
None

`delete_group`(*group*, *delete_volumes=False*)

Delete a group

Parameters

- **group** The *Group* to delete.
- **delete_volumes** (*bool*) When set to True, volumes in group will be deleted.

Returns
None.

`update_group`(*group*, ***attrs*)

Update a group

Parameters

- **group** The value can be the ID of a group or a *Group* instance.
- **attrs** (*dict*) The attributes to update on the group.

Returns
The updated group

Return type
Group

Group Snapshot Operations

class openstack.block_storage.v3._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

get_group_snapshot(*group_snapshot_id*)

Get a group snapshot

Parameters
group_snapshot_id The ID of the group snapshot to get.

Returns
A GroupSnapshot instance.

Return type
group_snapshot

find_group_snapshot(*name_or_id*, *ignore_missing=True*, ***, *details=True*)

Find a single group snapshot

Parameters

- **name_or_id** The name or ID of a group snapshot.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the group snapshot does not exist.
- **details** (*bool*) When set to `False`, no additional details will be returned. The default, `True`, will cause additional details to be returned.

Returns
One *group_snapshot*

Raises
NotFoundException when no resource can be found.

Raises
DuplicateResource when multiple resources are found.

group_snapshots(***, *details=True*, ***query*)

Retrieve a generator of group snapshots

Parameters

- **details** (*bool*) When `True`, returns *GroupSnapshot* objects with additional attributes filled.

- **query** (*kwargs*) Optional query parameters to be sent to limit the group snapshots being returned.

Returns
A generator of group snapshtos.

`create_group_snapshot`(***attrs*)

Create a group snapshot

Parameters
attrs (*dict*) Keyword arguments which will be used to create a *GroupSnapshot* comprised of the properties on the GroupSnapshot class.

Returns
The results of group snapshot creation.

Return type
group_snapshot.

`reset_group_snapshot_state`(*group_snapshot*, *state*)

Reset group snapshot status

Parameters

- **group_snapshot** The *GroupSnapshot* to set the state.
- **state** The state of the group snapshot to be set.

Returns
None

`delete_group_snapshot`(*group_snapshot*, *ignore_missing=True*)

Delete a group snapshot

Parameters
group_snapshot The GroupSnapshot to delete.

Returns
None

Group Type Operations

`class openstack.block_storage.v3._proxy.Proxy`(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

`get_group_type`(*group_type*)

Get a specific group type

Parameters
group_type The value can be the ID of a group type or a *GroupType* instance.

Returns
One :class: *~openstack.block_storage.v3.group_type.GroupType* instance.

Raises
NotFoundException when no resource can be found.

find_group_type(*name_or_id*, *ignore_missing=True*)

Find a single group type

Parameters

- **name_or_id** The name or ID of a group type.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the group type does not exist.

Returns
One *GroupType*

Raises
NotFoundException when no resource can be found.

Raises
DuplicateResource when multiple resources are found.

group_types(***query*)

Retrive a generator of group types

Parameters
query (*dict*) Optional query parameters to be sent to limit the resources being returned:

- sort: Comma-separated list of sort keys and optional sort directions in the form of <key> [:<direction>]. A valid direction is asc (ascending) or desc (descending).
- limit: Requests a page size of items. Returns a number of items up to a limit value. Use the limit parameter to make an initial limited request and use the ID of the last-seen item from the response as the marker parameter value in a subsequent limited request.
- offset: Used in conjunction with limit to return a slice of items. Is where to start in the list.
- marker: The ID of the last-seen item.

Returns
A generator of group type objects.

create_group_type(***attrs*)

Create a group type

Parameters
attrs (*dict*) Keyword arguments which will be used to create a *GroupType* comprised of the properties on the GroupType class.

Returns
The results of group type creation.

Return type
`GroupTye`.

delete_group_type(*group_type*, *ignore_missing=True*)

Delete a group type

Parameters

- **group_type** The value can be the ID of a group type or a *GroupType* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the zone does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent zone.

Returns

None

update_group_type(*group_type*, ***attrs*)

Update a group_type

Parameters

- **group_type** The value can be the ID of a group type or a *GroupType* instance.
- **attrs** (*dict*) The attributes to update on the group type.

Returns

The updated group type.

Return type

GroupType

fetch_group_type_group_specs(*group_type*)

Lists group specs of a group type.

Parameters

group_type Either the ID of a group type or a *GroupType* instance.

Returns

One *GroupType*

create_group_type_group_specs(*group_type*, *group_specs*)

Create group specs for a group type.

Parameters

- **group_type** Either the ID of a group type or a *GroupType* instance.
- **group_specs** (*dict*) dict of extra specs

Returns

One *GroupType*

get_group_type_group_specs_property(*group_type*, *prop*)

Retrieve a group spec property for a group type.

Parameters

- **group_type** Either the ID of a group type or a *GroupType* instance.
- **prop** (*str*) Property name.

Returns
String value of the requested property.

update_group_type_group_specs_property(*group_type*, *prop*, *val*)
Update a group spec property for a group type.

Parameters

- **group_type** Either the ID of a group type or a *GroupType* instance.
- **prop** (*str*) Property name.
- **val** (*str*) Property value.

Returns
String value of the requested property.

delete_group_type_group_specs_property(*group_type*, *prop*)
Delete a group spec property from a group type.

Parameters

- **group_type** Either the ID of a group type or a *GroupType* instance.
- **prop** (*str*) Property name.

Returns
None

Service Operations

class openstack.block_storage.v3._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

find_service(*name_or_id*, *ignore_missing=True*, ***query*)
Find a single service

Parameters

- **name_or_id** The name or ID of a service
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Additional attributes like host

Returns
One: class:*~openstack.block_storage.v3.service.Service* or None

Raises
NotFoundException when no resource can be found.

Raises
DuplicateResource when multiple resources are found.

services(***query*)

Return a generator of service

Parameters
query (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of Service objects

Return type
class: *~openstack.block_storage.v3.service.Service*

enable_service(*service*)

Enable a service

Parameters
service Either the ID of a service or a *Service* instance.

Returns
Updated service instance

Return type
class: *~openstack.block_storage.v3.service.Service*

disable_service(*service*, *, *reason=None*)

Disable a service

Parameters

- **service** Either the ID of a service or a *Service* instance
- **reason** (*str*) The reason to disable a service

Returns
Updated service instance

Return type
class: *~openstack.block_storage.v3.service.Service*

thaw_service(*service*)

Thaw a service

Parameters
service Either the ID of a service or a *Service* instance

Returns
Updated service instance

Return type
class: *~openstack.block_storage.v3.service.Service*

freeze_service(*service*)

Freeze a service

Parameters
service Either the ID of a service or a *Service* instance

Returns
Updated service instance

Return type
class: *~openstack.block_storage.v3.service.Service*

failover_service(*service*, *, *cluster=None*, *backend_id=None*)

Failover a service

Only applies to replicating cinder-volume services.

Parameters
service Either the ID of a service or a *Service* instance

Returns
Updated service instance

Return type
class: *~openstack.block_storage.v3.service.Service*

Type Operations

class openstack.block_storage.v3._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

get_type(*type*)

Get a single type

Parameters
type The value can be the ID of a type or a *Type* instance.

Returns
One *Type*

Raises
NotFoundException when no resource can be found.

find_type(*name_or_id*, *ignore_missing=True*)

Find a single volume type

Parameters

- **snapshot** The name or ID a volume type
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the volume type does not exist.

Returns
One *Type*

Raises
NotFoundException when no resource can be found.

Raises
DuplicateResource when multiple resources are found.

types(***query*)

Retrieve a generator of volume types

Returns
A generator of volume type objects.

create_type(***attrs*)

Create a new type from attributes

Parameters
attrs (*dict*) Keyword arguments which will be used to create a *Type*, comprised of the properties on the Type class.

Returns
The results of type creation

Return type
Type

delete_type(*type*, *ignore_missing=True*)

Delete a type

Parameters

- **type** The value can be either the ID of a type or a *Type* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the type does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent type.

Returns
None

update_type(*type*, ***attrs*)

Update a type

Parameters

- **type** The value can be either the ID of a type or a *Type* instance.
- **attrs** (*dict*) The attributes to update on the type

Returns
The updated type

Return type
Type

update_type_extra_specs(*type*, ***attrs*)

Update the extra_specs for a type

Parameters

- **type** The value can be either the ID of a type or a *Type* instance.
- **attrs** (*dict*) The extra spec attributes to update on the type

Returns
A dict containing updated extra_specs

delete_type_extra_specs(*type*, *keys*)

Delete the extra_specs for a type

Note: This method will do a HTTP DELETE request for every key in keys.

Parameters

- **type** The value can be either the ID of a type or a *Type* instance.
- **keys** The keys to delete

Returns
None

get_type_access(*type*)

Lists project IDs that have access to private volume type.

Parameters
type The value can be either the ID of a type or a *Type* instance.

Returns
List of dictionaries describing projects that have access to the specified type

add_type_access(*type*, *project_id*)

Adds private volume type access to a project.

Parameters

- **type** The value can be either the ID of a type or a *Type* instance.
- **project_id** (*str*) The ID of the project. Volume Type access to be added to this project ID.

Returns
None

remove_type_access(*type*, *project_id*)

Remove private volume type access from a project.

Parameters

- **type** The value can be either the ID of a type or a *Type* instance.
- **project_id** (*str*) The ID of the project. Volume Type access to be removed to this project ID.

Returns
None

get_type_encryption(*volume_type_id*)

Get the encryption details of a volume type

Parameters
volume_type_id The value can be the ID of a type or a *Type* instance.

Returns
One *TypeEncryption*

Raises
NotFoundException when no resource can be found.

create_type_encryption(*volume_type*, ***attrs*)

Create new type encryption from attributes

Parameters

- **volume_type** The value can be the ID of a type or a *Type* instance.
- **attrs** (*dict*) Keyword arguments which will be used to create a *TypeEncryption*, comprised of the properties on the TypeEncryption class.

Returns

The results of type encryption creation

Return type

TypeEncryption

delete_type_encryption(*encryption=None*, *volume_type=None*, *ignore_missing=True*)

Delete type encryption attributes

Parameters

- **encryption** The value can be None or a *TypeEncryption* instance. If encryption_id is None then volume_type_id must be specified.
- **volume_type** The value can be the ID of a type or a *Type* instance. Required if encryption_id is None.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the type does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent type.

Returns

None

update_type_encryption(*encryption=None*, *volume_type=None*, ***attrs*)

Update a type

Parameters

- **encryption** The value can be None or a *TypeEncryption* instance. If this is `None` then `volume_type_id` must be specified.
- **volume_type** The value can be the ID of a type or a *Type* instance. Required if `encryption_id` is None.
- **attrs** (*dict*) The attributes to update on the type encryption.

Returns

The updated type encryption

Return type

TypeEncryption

Snapshot Operations

class openstack.block_storage.v3._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

get_snapshot(*snapshot*)

Get a single snapshot

Parameters

snapshot The value can be the ID of a snapshot or a *Snapshot* instance.

Returns

One *Snapshot*

Raises

NotFoundException when no resource can be found.

find_snapshot(*name_or_id*, *ignore_missing=True*, ***, *details=True*, *all_projects=False*)

Find a single snapshot

Parameters

- **snapshot** The name or ID a snapshot
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the snapshot does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **details** (*bool*) When set to `False` :class: *~openstack.block_storage.v3.snapshot.Snapshot* objects will be returned. The default, `True`, will cause more attributes to be returned.
- **all_projects** (*bool*) When set to `True`, search for snapshot by name across all projects. Note that this will likely result in a higher chance of duplicates. Admin-only by default.

Returns

One *Snapshot*

Raises

NotFoundException when no resource can be found.

Raises

DuplicateResource when multiple resources are found.

snapshots(***, *details=True*, *all_projects=False*, ***query*)

Retrieve a generator of snapshots

Parameters

- **details** (*bool*) When set to `False` :class: *~openstack.block_storage.v3.snapshot.Snapshot* objects will be returned. The default, `True`, will cause more attributes to be returned.
- **all_projects** (*bool*) When set to `True`, list snapshots from all projects. Admin-only by default.
- **query** (*kwargs*) Optional query parameters to be sent to limit the snapshots being returned. Available parameters include:
  - name: Name of the snapshot as a string.
  - project_id: Filter the snapshots by project.

- volume_id: volume id of a snapshot.
- status: Value of the status of the snapshot so that you can filter on available for example.

Returns
A generator of snapshot objects.

`create_snapshot`(***attrs*)

Create a new snapshot from attributes

Parameters
attrs (*dict*) Keyword arguments which will be used to create a *Snapshot*, comprised of the properties on the Snapshot class.

Returns
The results of snapshot creation

Return type
Snapshot

`update_snapshot`(*snapshot*, ***attrs*)

Update a snapshot

Parameters

- **snapshot** Either the ID of a snapshot or a *Snapshot* instance.
- **attrs** (*dict*) The attributes to update on the snapshot.

Returns
The updated snapshot

Return type
Snapshot

`delete_snapshot`(*snapshot*, *ignore_missing=True*, *force=False*)

Delete a snapshot

Parameters

- **snapshot** The value can be either the ID of a snapshot or a *Snapshot* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the snapshot does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent snapshot.
- **force** (*bool*) Whether to try forcing snapshot deletion.

Returns
`None`

`get_snapshot_metadata`(*snapshot*)

Return a dictionary of metadata for a snapshot

Parameters
snapshot Either the ID of a snapshot or a *Snapshot*.

Returns
A *Snapshot* with the snapshots metadata. All keys and values are Unicode text.

Return type
Snapshot

set_snapshot_metadata(*snapshot*, ***metadata*)

Update metadata for a snapshot

Parameters

- **snapshot** Either the ID of a snapshot or a *Snapshot*.
- **metadata** (*kwargs*) Key/value pairs to be updated in the snapshots metadata. No other metadata is modified by this call. All keys and values are stored as Unicode.

Returns
A *Snapshot* with the snapshots metadata. All keys and values are Unicode text.

Return type
Snapshot

delete_snapshot_metadata(*snapshot*, *keys=None*)

Delete metadata for a snapshot

Parameters

- **snapshot** Either the ID of a snapshot or a *Snapshot*.
- **keys** (*list*) The keys to delete. If left empty complete metadata will be removed.

Return type
None

reset_snapshot(*snapshot*, *status*)

Reset status of the snapshot

Parameters

- **snapshot** The value can be either the ID of a backup or a *Snapshot* instance.
- **status** (*str*) New snapshot status

Returns
None

set_snapshot_status(*snapshot*, *status*, *progress=None*)

Update fields related to the status of a snapshot.

Parameters

- **snapshot** The value can be either the ID of a backup or a *Snapshot* instance.
- **status** (*str*) New snapshot status

- **progress** (*str*) A percentage value for snapshot build progress.

Returns
None

manage_snapshot(***attrs*)

Creates a snapshot by using existing storage rather than allocating new storage.

Parameters
attrs (*dict*) Keyword arguments which will be used to create a *Snapshot*, comprised of the properties on the Snapshot class.

Returns
The results of snapshot creation

Return type
Snapshot

unmanage_snapshot(*snapshot*)

Unmanage a snapshot from block storage provisioning.

Parameters
snapshot Either the ID of a snapshot or a *Snapshot*.

Returns
None

Stats Operations

class openstack.block_storage.v3._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

backend_pools(***query*)

Returns a generator of cinder Back-end storage pools

Parameters
query (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

:returns A generator of cinder Back-end storage pools objects

QuotaClassSet Operations

class openstack.block_storage.v3._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

get_quota_class_set(*quota_class_set='default'*)

Get a single quota class set

Only one quota class is permitted, `default`.

Parameters

quota_class_set The value can be the ID of a quota class set (only `default` is supported) or a `QuotaClassSet` instance.

Returns

One `QuotaClassSet`

Raises

NotFoundException when no resource can be found.

update_quota_class_set(*quota_class_set*, ***attrs*)

Update a QuotaClassSet.

Only one quota class is permitted, `default`.

Parameters

- **quota_class_set** Either the ID of a quota class set (only `default` is supported) or a
- **attrs** The attributes to update on the QuotaClassSet represented by `quota_class_set`.

Returns

The updated QuotaSet

Return type

QuotaSet

QuotaSet Operations

class openstack.block_storage.v3._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

get_quota_set(*project*, *usage=False*, ***query*)

Show QuotaSet information for the project

Parameters

- **project** ID or instance of `Project` of the project for which the quota should be retrieved
- **usage** (*bool*) When set to `True` quota usage and reservations would be filled.
- **query** (*dict*) Additional query parameters to use.

Returns

One *QuotaSet*

Raises
NotFoundException when no resource can be found.

get_quota_set_defaults(*project*)

Show QuotaSet defaults for the project

Parameters
project ID or instance of `Project` of the project for which the quota should be retrieved

Returns
One *QuotaSet*

Raises
NotFoundException when no resource can be found.

revert_quota_set(*project*, ***query*)

Reset Quota for the project/user.

Parameters

- **project** ID or instance of `Project` of the project for which the quota should be resetted.
- **query** (*dict*) Additional parameters to be used.

Returns
`None`

update_quota_set(*project*, ***attrs*)

Update a QuotaSet.

Parameters

- **project** ID or instance of `Project` of the project for which the quota should be reset.
- **attrs** The attributes to update on the QuotaSet represented by `quota_set`.

Returns
The updated QuotaSet

Return type
QuotaSet

BlockStorageSummary Operations

class openstack.block_storage.v3._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

summary(*all_projects*, ***kwargs*)

Get Volumes Summary

This method returns the volumes summary in the deployment.

Parameters
all_projects Whether to return the summary of all projects or not.

Returns
One :class: *~openstack.block_storage.v3.block_storage_summary.Summary* instance.

Attachments

class openstack.block_storage.v3._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_attachment(*volume*, ***attrs*)

Create a new attachment

This is an internal API and should only be called by services consuming volume attachments like nova, glance, ironic etc.

Parameters

- **volume** The value can be either the ID of a volume or a *Volume* instance.
- **attrs** (*dict*) Keyword arguments which will be used to create a *Attachment* comprised of the properties on the Attachment class like connector, instance_id, mode etc.

Returns
The results of attachment creation

Return type
Attachment

get_attachment(*attachment*)

Get a single volume

This is an internal API and should only be called by services consuming volume attachments like nova, glance, ironic etc.

Parameters
attachment The value can be the ID of an attachment or a `Attachment` instance.

Returns
One `Attachment`

Raises
NotFoundException when no resource can be found.

attachments(***query*)

Returns a generator of attachments.

This is an internal API and should only be called by services consuming volume attachments like nova, glance, ironic etc.

Parameters
query (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of attachment objects.

delete_attachment(*attachment*, *ignore_missing=True*)

Delete an attachment

This is an internal API and should only be called by services consuming volume attachments like nova, glance, ironic etc.

Parameters

- **type** The value can be either the ID of a attachment or a *Attachment* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the attachment does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent attachment.

Returns
`None`

update_attachment(*attachment*, ***attrs*)

Update an attachment

This is an internal API and should only be called by services consuming volume attachments like nova, glance, ironic etc.

Parameters

- **attachment** The value can be the ID of an attachment or a *Attachment* instance.
- **attrs** (*dict*) Keyword arguments which will be used to update a *Attachment* comprised of the properties on the Attachment class

Returns
The updated attachment

Return type
`Attachment`

complete_attachment(*attachment*)

Complete an attachment

This is an internal API and should only be called by services consuming volume attachments like nova, glance, ironic etc.

Parameters
attachment The value can be the ID of an attachment or a *Attachment* instance.

Returns
`None`

Return type
`Attachment`

Transfer Operations

class openstack.block_storage.v3._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_transfer(***attrs*)

Create a new Transfer record

Parameters

- **volume_id** The value is ID of the volume.
- **name** The value is name of the transfer
- **attrs** (*dict*) Keyword arguments which will be used to create a *Transfer* comprised of the properties on the Transfer class.

Returns

The results of Transfer creation

Return type

Transfer

delete_transfer(*transfer*, *ignore_missing=True*)

Delete a volume transfer

Parameters

- **transfer** The value can be either the ID of a transfer or a `Transfer` instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the transfer does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent transfer.

Returns

None

find_transfer(*name_or_id*, *ignore_missing=True*)

Find a single transfer

Parameters

- **name_or_id** The name or ID a transfer
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the volume transfer does not exist.

Returns

One *Transfer*

Raises

NotFoundException when no resource can be found.

Raises

DuplicateResource when multiple resources are found.

`get_transfer`(*transfer*)

Get a single transfer

Parameters

transfer The value can be the ID of a transfer or a *Transfer* instance.

Returns

One *Transfer*

Raises

NotFoundException when no resource can be found.

`transfers`(*, *details=True*, *all_projects=False*, ***query*)

Retrieve a generator of transfers

Parameters

- **details** (*bool*) When set to `False` no extended attributes will be returned. The default, `True`, will cause objects with additional attributes to be returned.
- **all_projects** (*bool*) When set to `True`, list transfers from all projects. Admin-only by default.
- **query** (*kwargs*) Optional query parameters to be sent to limit the transfers being returned.

Returns

A generator of transfer objects.

`accept_transfer`(*transfer_id*, *auth_key*)

Accept a Transfer

Parameters

- **transfer_id** The value can be the ID of a transfer or a *Transfer* instance.
- **auth_key** The key to authenticate volume transfer.

Returns

The results of Transfer creation

Return type

Transfer

Helpers

`class openstack.block_storage.v3._proxy.Proxy`(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

`wait_for_status`(*res*, *status='available'*, *failures=None*, *interval=2*, *wait=None*, *attribute='status'*, *callback=None*)

Wait for the resource to be in a particular status.

Parameters

- **session** The session to use for making this request.
- **resource** The resource to wait on to reach the status. The resource must have a status attribute specified via `attribute`.
- **status** Desired status of the resource.
- **failures** Statuses that would indicate the transition failed such as ERROR. Defaults to [ERROR].
- **interval** Number of seconds to wait between checks.
- **wait** Maximum number of seconds to wait for transition. Set to `None` to wait forever.
- **attribute** Name of the resource attribute that contains the status.
- **callback** A callback function. This will be called with a single value, progress. This is API specific but is generally a percentage value from 0-100.

Returns
The updated resource.

Raises
ResourceTimeout if the transition to status failed to occur in `wait` seconds.

Raises
ResourceFailure if the resource transitioned to one of the states in `failures`.

Raises
`AttributeError` if the resource does not have a `status` attribute

wait_for_delete(*res*, *interval=2*, *wait=120*, *callback=None*)

Wait for a resource to be deleted.

Parameters

- **res** The resource to wait on to be deleted.
- **interval** Number of seconds to wait before to consecutive checks.
- **wait** Maximum number of seconds to wait before the change.
- **callback** A callback function. This will be called with a single value, progress, which is a percentage value from 0-100.

Returns
The resource is returned on success.

Raises
ResourceTimeout if transition to delete failed to occur in the specified seconds.

Default Volume Types

class openstack.block_storage.v3._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

default_types()

Lists default types.

Returns
List of default types associated to projects.

show_default_type(*project*)

Show default type for a project.

Parameters
project The value can be either the ID of a project or a *Project* instance.

Returns
Default type associated to the project.

set_default_type(*project*, *type*)

Set default type for a project.

Parameters

- **project** The value can be either the ID of a project or a *Project* instance.
- **type** The value can be either the ID of a type or a *Type* instance.

Returns
Dictionary of project ID and its associated default type.

unset_default_type(*project*)

Unset default type for a project.

Parameters
project The value can be either the ID of a project or a *Project* instance.

Returns
None

Cluster API

The Cluster Class

The cluster high-level interface is available through the `cluster` member of a *Connection* object. The `cluster` member will only be added if the service is detected.

Build Info Operations

class openstack.clustering.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

get_build_info()

Get build info for service engine and API

Returns
A dictionary containing the API and engine revision string.

Profile Type Operations

class openstack.clustering.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

profile_types(***query*)

Get a generator of profile types.

Returns
A generator of objects that are of type *ProfileType*

get_profile_type(*profile_type*)

Get the details about a profile type.

Parameters
profile_type The name of the profile_type to retrieve or an object of *ProfileType*.

Returns
A *ProfileType* object.

Raises
NotFoundException when no profile_type matching the name could be found.

Profile Operations

class openstack.clustering.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_profile(***attrs*)

Create a new profile from attributes.

Parameters
attrs (*dict*) Keyword arguments that will be used to create a *Profile*, it is comprised of the properties on the Profile class.

Returns
The results of profile creation.

Return type
Profile.

delete_profile(*profile*, *ignore_missing=True*)

Delete a profile.

Parameters

- **profile** The value can be either the name or ID of a profile or a *Profile* instance.
- **ignore_missing** (*bool*) When set to `False`, an exception *NotFoundException* will be raised when the profile could not be found. When set to `True`, no exception will be raised when attempting to delete a non-existent profile.

Returns
`None`

find_profile(*name_or_id*, *ignore_missing=True*)

Find a single profile.

Parameters

- **name_or_id** (*str*) The name or ID of a profile.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns
One *Profile* object or None

get_profile(*profile*)

Get a single profile.

Parameters
profile The value can be the name or ID of a profile or a *Profile* instance.

Returns
One *Profile*

Raises
NotFoundException when no profile matching the criteria could be found.

profiles(***query*)

Retrieve a generator of profiles.

Parameters
query (*kwargs*) Optional query parameters to be sent to restrict the profiles to be returned. Available parameters include:

- name: The name of a profile.
- type: The type name of a profile.
- metadata: A list of key-value pairs that are associated with a profile.
- sort: A list of sorting keys separated by commas. Each sorting key can optionally be attached with a sorting direction modifier which can be `asc` or `desc`.

- limit: Requests a specified size of returned items from the query. Returns a number of items up to the specified limit value.
- marker: Specifies the ID of the last-seen item. Use the limit parameter to make an initial limited request and use the ID of the last-seen item from the response as the marker parameter value in a subsequent limited request.
- global_project: A boolean value indicating whether profiles from all projects will be returned.

Returns
A generator of profile instances.

update_profile(*profile*, ***attrs*)

Update a profile.

Parameters

- **profile** Either the name or the ID of the profile, or an instance of *Profile*.
- **attrs** The attributes to update on the profile represented by the `value` parameter.

Returns
The updated profile.

Return type
Profile

validate_profile(***attrs*)

Validate a profile spec.

Parameters
attrs (*dict*) Keyword arguments that will be used to create a `ProfileValidate`, it is comprised of the properties on the Profile class.

Returns
The results of profile validation.

Return type
`ProfileValidate`.

Policy Type Operations

class openstack.clustering.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

policy_types(***query*)

Get a generator of policy types.

Returns
A generator of objects that are of type *PolicyType*

get_policy_type(*policy_type*)

Get the details about a policy type.

Parameters

policy_type The name of a poicy_type or an object of *PolicyType*.

Returns

A *PolicyType* object.

Raises

NotFoundException when no policy_type matching the name could be found.

Policy Operations

class openstack.clustering.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_policy(***attrs*)

Create a new policy from attributes.

Parameters

attrs (*dict*) Keyword arguments that will be used to create a *Policy*, it is comprised of the properties on the `Policy` class.

Returns

The results of policy creation.

Return type

Policy.

delete_policy(*policy*, *ignore_missing=True*)

Delete a policy.

Parameters

- **policy** The value can be either the name or ID of a policy or a *Policy* instance.
- **ignore_missing** (*bool*) When set to `False`, an exception *NotFoundException* will be raised when the policy could not be found. When set to `True`, no exception will be raised when attempting to delete a non-existent policy.

Returns

`None`

find_policy(*name_or_id*, *ignore_missing=True*)

Find a single policy.

Parameters

- **name_or_id** (*str*) The name or ID of a policy.

- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the specified policy does not exist. When set to `True`, None will be returned when attempting to find a nonexistent policy.

Returns
A policy object or None.

Return type
Policy

get_policy(*policy*)

Get a single policy.

Parameters
policy The value can be the name or ID of a policy or a *Policy* instance.

Returns
A policy object.

Return type
Policy

Raises
NotFoundException when no policy matching the criteria could be found.

policies(***query*)

Retrieve a generator of policies.

Parameters
query (*kwargs*) Optional query parameters to be sent to restrict the policies to be returned. Available parameters include:

- name: The name of a policy.
- type: The type name of a policy.
- sort: A list of sorting keys separated by commas. Each sorting key can optionally be attached with a sorting direction modifier which can be `asc` or `desc`.
- limit: Requests a specified size of returned items from the query. Returns a number of items up to the specified limit value.
- marker: Specifies the ID of the last-seen item. Use the limit parameter to make an initial limited request and use the ID of the last-seen item from the response as the marker parameter value in a subsequent limited request.
- global_project: A boolean value indicating whether policies from all projects will be returned.

Returns
A generator of policy instances.

update_policy(*policy*, ***attrs*)

Update a policy.

Parameters

- **policy** Either the name or the ID of a policy, or an instance of *Policy*.

- **attrs** The attributes to update on the policy represented by the `value` parameter.

Returns
The updated policy.

Return type
Policy

validate_policy

Cluster Operations

class openstack.clustering.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_cluster(***attrs*)

Create a new cluster from attributes.

Parameters
attrs (*dict*) Keyword arguments that will be used to create a *Cluster*, it is comprised of the properties on the Cluster class.

Returns
The results of cluster creation.

Return type
Cluster.

delete_cluster(*cluster*, *ignore_missing=True*, *force_delete=False*)

Delete a cluster.

Parameters

- **cluster** The value can be either the name or ID of a cluster or a `Cluster` instance.
- **ignore_missing** (*bool*) When set to `False`, an exception *NotFoundException* will be raised when the cluster could not be found. When set to `True`, no exception will be raised when attempting to delete a non-existent cluster.
- **force_delete** (*bool*) When set to `True`, the cluster deletion will be forced immediately.

Returns
The instance of the Cluster which was deleted.

Return type
`Cluster`.

find_cluster(*name_or_id*, *ignore_missing=True*)

Find a single cluster.

Parameters

- **name_or_id** (*str*) The name or ID of a cluster.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns
One *Cluster* object or None

get_cluster(*cluster*)

Get a single cluster.

Parameters
cluster The value can be the name or ID of a cluster or a *Cluster* instance.

Returns
One *Cluster*

Raises
NotFoundException when no cluster matching the criteria could be found.

clusters(***query*)

Retrieve a generator of clusters.

Parameters
query (*kwargs*) Optional query parameters to be sent to restrict the clusters to be returned. Available parameters include:

- name: The name of a cluster.
- status: The current status of a cluster.
- sort: A list of sorting keys separated by commas. Each sorting key can optionally be attached with a sorting direction modifier which can be `asc` or `desc`.
- limit: Requests a specified size of returned items from the query. Returns a number of items up to the specified limit value.
- marker: Specifies the ID of the last-seen item. Use the limit parameter to make an initial limited request and use the ID of the last-seen item from the response as the marker parameter value in a subsequent limited request.
- global_project: A boolean value indicating whether clusters from all projects will be returned.

Returns
A generator of cluster instances.

update_cluster(*cluster*, ***attrs*)

Update a cluster.

Parameters

- **cluster** Either the name or the ID of the cluster, or an instance of *Cluster*.
- **attrs** The attributes to update on the cluster represented by the `cluster` parameter.

Returns
The updated cluster.

Return type
Cluster

add_nodes_to_cluster(*cluster*, *nodes*)

Add nodes to a cluster.

Parameters

- **cluster** Either the name or the ID of the cluster, or an instance of *Cluster*.
- **nodes** List of nodes to be added to the cluster.

Returns
A dict containing the action initiated by this operation.

remove_nodes_from_cluster(*cluster*, *nodes*, ***params*)

Remove nodes from a cluster.

Parameters

- **cluster** Either the name or the ID of the cluster, or an instance of *Cluster*.
- **nodes** List of nodes to be removed from the cluster.
- **params** (*kwargs*) Optional query parameters to be sent to restrict the nodes to be returned. Available parameters include:
  - destroy_after_deletion: A boolean value indicating whether the deleted nodes to be destroyed right away.

Returns
A dict containing the action initiated by this operation.

replace_nodes_in_cluster(*cluster*, *nodes*)

Replace the nodes in a cluster with specified nodes.

Parameters

- **cluster** Either the name or the ID of the cluster, or an instance of *Cluster*.
- **nodes** List of nodes to be deleted/added to the cluster.

Returns
A dict containing the action initiated by this operation.

scale_out_cluster(*cluster*, *count=None*)

Inflate the size of a cluster.

Parameters

- **cluster** Either the name or the ID of the cluster, or an instance of *Cluster*.
- **count** Optional parameter specifying the number of nodes to be added.

Returns
A dict containing the action initiated by this operation.

scale_in_cluster(*cluster*, *count=None*)

Shrink the size of a cluster.

Parameters

- **cluster** Either the name or the ID of the cluster, or an instance of *Cluster*.
- **count** Optional parameter specifying the number of nodes to be removed.

Returns

A dict containing the action initiated by this operation.

resize_cluster(*cluster*, ***params*)

Resize of cluster.

Parameters

- **cluster** Either the name or the ID of the cluster, or an instance of *Cluster*.
- **params** (*dict*) A dictionary providing the parameters for the resize action.

Returns

A dict containing the action initiated by this operation.

attach_policy_to_cluster(*cluster*, *policy*, ***params*)

Attach a policy to a cluster.

Parameters

- **cluster** Either the name or the ID of the cluster, or an instance of *Cluster*.
- **policy** Either the name or the ID of a policy.
- **params** (*dict*) A dictionary containing the properties for the policy to be attached.

Returns

A dict containing the action initiated by this operation.

detach_policy_from_cluster(*cluster*, *policy*)

Detach a policy from a cluster.

Parameters

- **cluster** Either the name or the ID of the cluster, or an instance of *Cluster*.
- **policy** Either the name or the ID of a policy.

Returns

A dict containing the action initiated by this operation.

update_cluster_policy(*cluster*, *policy*, ***params*)

Change properties of a policy which is bound to the cluster.

Parameters

- **cluster** Either the name or the ID of the cluster, or an instance of *Cluster*.
- **policy** Either the name or the ID of a policy.
- **params** (*dict*) A dictionary containing the new properties for the policy.

Returns
A dict containing the action initiated by this operation.

`collect_cluster_attrs`(*cluster*, *path*, ***query*)

Collect attribute values across a cluster.

Parameters

- **cluster** The value can be either the ID of a cluster or a *Cluster* instance.
- **path** A Json path string specifying the attribute to collect.
- **query** Optional query parameters to be sent to limit the resources being returned.

Returns
A dictionary containing the list of attribute values.

`check_cluster`(*cluster*, ***params*)

Check a cluster.

Parameters

- **cluster** The value can be either the ID of a cluster or a *Cluster* instance.
- **params** (*dict*) A dictionary providing the parameters for the check action.

Returns
A dictionary containing the action ID.

`recover_cluster`(*cluster*, ***params*)

recover a cluster.

Parameters

- **cluster** The value can be either the ID of a cluster or a *Cluster* instance.
- **params** (*dict*) A dictionary providing the parameters for the recover action.

Returns
A dictionary containing the action ID.

`perform_operation_on_cluster`(*cluster*, *operation*, ***params*)

Perform an operation on the specified cluster.

Parameters

- **cluster** The value can be either the ID of a cluster or a *Cluster* instance.
- **operation** A string specifying the operation to be performed.
- **params** (*dict*) A dictionary providing the parameters for the operation.

Returns
A dictionary containing the action ID.

`cluster_policies`(*cluster*, ***query*)

Retrieve a generator of cluster-policy bindings.

Parameters

- **cluster** The value can be the name or ID of a cluster or a *Cluster* instance.

- **query** (*kwargs*) Optional query parameters to be sent to restrict the policies to be returned. Available parameters include:
  - enabled: A boolean value indicating whether the policy is enabled on the cluster.

Returns
A generator of cluster-policy binding instances.

`get_cluster_policy`(*cluster_policy*, *cluster*)

Get a cluster-policy binding.

Parameters

- **cluster_policy** The value can be the name or ID of a policy or a *Policy* instance.
- **cluster** The value can be the name or ID of a cluster or a *Cluster* instance.

Returns
a cluster-policy binding object.

Return type
CLusterPolicy

Raises
NotFoundException when no cluster-policy binding matching the criteria could be found.

Node Operations

`class openstack.clustering.v1._proxy.Proxy`(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

`create_node`(***attrs*)

Create a new node from attributes.

Parameters
attrs (*dict*) Keyword arguments that will be used to create a *Node*, it is comprised of the properties on the Node class.

Returns
The results of node creation.

Return type
Node.

`delete_node`(*node*, *ignore_missing=True*, *force_delete=False*)

Delete a node.

Parameters

- **node** The value can be either the name or ID of a node or a Node instance.

- **ignore_missing** (*bool*) When set to `False`, an exception *NotFoundException* will be raised when the node could not be found. When set to `True`, no exception will be raised when attempting to delete a non-existent node.
- **force_delete** (*bool*) When set to `True`, the node deletion will be forced immediately.

Returns
The instance of the Node which was deleted.

Return type
`Node`.

find_node(*name_or_id*, *ignore_missing=True*)

Find a single node.

Parameters

- **name_or_id** (*str*) The name or ID of a node.
- **ignore_missing** (*bool*) When set to False *NotFoundException* will be raised when the specified node does not exist. when set to True, None will be returned when attempting to find a nonexistent policy

Returns
One *Node* object or None.

get_node(*node*, *details=False*)

Get a single node.

Parameters

- **node** The value can be the name or ID of a node or a *Node* instance.
- **details** An optional argument that indicates whether the server should return more details when retrieving the node data.

Returns
One *Node*

Raises
NotFoundException when no node matching the name or ID could be found.

nodes(***query*)

Retrieve a generator of nodes.

Parameters
query (*kwargs*) Optional query parameters to be sent to restrict the nodes to be returned. Available parameters include:

- cluster_id: A string including the name or ID of a cluster to which the resulted node(s) is a member.
- name: The name of a node.
- status: The current status of a node.
- sort: A list of sorting keys separated by commas. Each sorting key can optionally be attached with a sorting direction modifier which can be `asc` or `desc`.

- limit: Requests at most the specified number of items be returned from the query.
- marker: Specifies the ID of the last-seen node. Use the limit parameter to make an initial limited request and use the ID of the last-seen node from the response as the marker parameter value in a subsequent limited request.
- global_project: A boolean value indicating whether nodes from all projects will be returned.

Returns
A generator of node instances.

update_node(*node*, ***attrs*)

Update a node.

Parameters

- **node** Either the name or the ID of the node, or an instance of *Node*.
- **attrs** The attributes to update on the node represented by the node parameter.

Returns
The updated node.

Return type
Node

check_node(*node*, ***params*)

Check the health of the specified node.

Parameters

- **node** The value can be either the ID of a node or a *Node* instance.
- **params** (*dict*) A dictionary providing the parametes to the check action.

Returns
A dictionary containing the action ID.

recover_node(*node*, ***params*)

Recover the specified node into healthy status.

Parameters

- **node** The value can be either the ID of a node or a *Node* instance.
- **params** (*dict*) A dict supplying parameters to the recover action.

Returns
A dictionary containing the action ID.

adopt_node(*preview=False*, ***attrs*)

Adopting an existing resource as a node.

Parameters

- **preview** A boolean indicating whether this is a preview operation which means only the profile to be used is returned rather than creating a node object using that profile.

- **attrs** (*dict*) Keyword parameters for node adoption. Valid parameters include:
  - type: (Required) A string containing the profile type and version to be used for node adoption. For example, `os.nova.sever-1.0`.
  - identity: (Required) A string including the name or ID of an OpenStack resource to be adopted as a Senlin node.
  - name: (Optional) The name of node to be created. Omitting this parameter will have the node named automatically.
  - snapshot: (Optional) A boolean indicating whether a snapshot of the target resource should be created if possible. Default is False.
  - metadata: (Optional) A dictionary of arbitrary key-value pairs to be associated with the adopted node.
  - overrides: (Optional) A dictionary of key-value pairs to be used to override attributes derived from the target resource.

Returns
The result of node adoption. If *preview* is set to False (default), returns a *Node* object, otherwise a Dict is returned containing the profile to be used for the new node.

perform_operation_on_node(*node*, *operation*, ***params*)

Perform an operation on the specified node.

Parameters

- **node** The value can be either the ID of a node or a *Node* instance.
- **operation** A string specifying the operation to be performed.
- **params** (*dict*) A dictionary providing the parameters for the operation.

Returns
A dictionary containing the action ID.

Receiver Operations

class openstack.clustering.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_receiver(***attrs*)

Create a new receiver from attributes.

Parameters
attrs (*dict*) Keyword arguments that will be used to create a *Receiver*, it is comprised of the properties on the Receiver class.

Returns
The results of receiver creation.

Return type
Receiver.

update_receiver(*receiver*, ***attrs*)

Update a receiver.

Parameters

- **receiver** The value can be either the name or ID of a receiver or a *Receiver* instance.
- **attrs** The attributes to update on the receiver parameter. Valid attribute names include `name`, `action` and `params`.

Returns
The updated receiver.

Return type
Receiver

delete_receiver(*receiver*, *ignore_missing=True*)

Delete a receiver.

Parameters

- **receiver** The value can be either the name or ID of a receiver or a *Receiver* instance.
- **ignore_missing** (*bool*) When set to `False`, an exception *NotFoundException* will be raised when the receiver could not be found. When set to `True`, no exception will be raised when attempting to delete a non-existent receiver.

Returns
`None`

find_receiver(*name_or_id*, *ignore_missing=True*)

Find a single receiver.

Parameters

- **name_or_id** (*str*) The name or ID of a receiver.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the specified receiver does not exist. When set to `True`, None will be returned when attempting to find a nonexistent receiver.

Returns
A receiver object or None.

Return type
Receiver

get_receiver(*receiver*)

Get a single receiver.

Parameters
receiver The value can be the name or ID of a receiver or a *Receiver* instance.

Returns
A receiver object.

Return type
Receiver

Raises
NotFoundException when no receiver matching the criteria could be found.

receivers(***query*)

Retrieve a generator of receivers.

Parameters
query (*kwargs*) Optional query parameters for restricting the receivers to be returned. Available parameters include:

- name: The name of a receiver object.
- type: The type of receiver objects.
- cluster_id: The ID of the associated cluster.
- action: The name of the associated action.
- sort: A list of sorting keys separated by commas. Each sorting key can optionally be attached with a sorting direction modifier which can be `asc` or `desc`.
- global_project: A boolean value indicating whether receivers
- from all projects will be returned.

Returns
A generator of receiver instances.

Action Operations

class openstack.clustering.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

get_action(*action*)

Get a single action.

Parameters
action The value can be the name or ID of an action or a *Action* instance.

Returns
an action object.

Return type
Action

Raises
NotFoundException when no action matching the criteria could be found.

actions(***query*)

Retrieve a generator of actions.

Parameters

query (*kwargs*) Optional query parameters to be sent to restrict the actions to be returned. Available parameters include:

- name: name of action for query.
- target: ID of the target object for which the actions should be returned.
- action: built-in action types for query.
- sort: A list of sorting keys separated by commas. Each sorting key can optionally be attached with a sorting direction modifier which can be `asc` or `desc`.
- limit: Requests a specified size of returned items from the query. Returns a number of items up to the specified limit value.
- marker: Specifies the ID of the last-seen item. Use the limit parameter to make an initial limited request and use the ID of the last-seen item from the response as the marker parameter value in a subsequent limited request.

Returns

A generator of action instances.

Event Operations

class openstack.clustering.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

get_event(*event*)

Get a single event.

Parameters

event The value can be the name or ID of an event or a *Event* instance.

Returns

an event object.

Return type

Event

Raises

NotFoundException when no event matching the criteria could be found.

events(***query*)

Retrieve a generator of events.

Parameters

query (*kwargs*) Optional query parameters to be sent to restrict the events to be returned. Available parameters include:

- obj_name: name string of the object associated with an event.
- obj_type: type string of the object related to an event. The value can be `cluster`, `node`, `policy` etc.
- obj_id: ID of the object associated with an event.
- cluster_id: ID of the cluster associated with the event, if any.
- action: name of the action associated with an event.
- sort: A list of sorting keys separated by commas. Each sorting key can optionally be attached with a sorting direction modifier which can be `asc` or `desc`.
- limit: Requests a specified size of returned items from the query. Returns a number of items up to the specified limit value.
- marker: Specifies the ID of the last-seen item. Use the limit parameter to make an initial limited request and use the ID of the last-seen item from the response as the marker parameter value in a subsequent limited request.
- global_project: A boolean specifying whether events from all projects should be returned. This option is subject to access control checking.

Returns

A generator of event instances.

Helper Operations

class openstack.clustering.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

wait_for_status(*res*, *status*, *failures=None*, *interval=2*, *wait=None*, *attribute='status'*, *callback=None*)

Wait for the resource to be in a particular status.

Parameters

- **session** The session to use for making this request.
- **resource** The resource to wait on to reach the status. The resource must have a status attribute specified via `attribute`.
- **status** Desired status of the resource.
- **failures** Statuses that would indicate the transition failed such as ERROR. Defaults to [ERROR].
- **interval** Number of seconds to wait between checks.
- **wait** Maximum number of seconds to wait for transition. Set to `None` to wait forever.
- **attribute** Name of the resource attribute that contains the status.

- **callback** A callback function. This will be called with a single value, progress. This is API specific but is generally a percentage value from 0-100.

Returns
The updated resource.

Raises
ResourceTimeout if the transition to status failed to occur in `wait` seconds.

Raises
ResourceFailure if the resource transitioned to one of the states in `failures`.

Raises
`AttributeError` if the resource does not have a `status` attribute

wait_for_delete(*res*, *interval=2*, *wait=120*, *callback=None*)

Wait for a resource to be deleted.

Parameters

- **res** The resource to wait on to be deleted.
- **interval** Number of seconds to wait before to consecutive checks.
- **wait** Maximum number of seconds to wait before the change.
- **callback** A callback function. This will be called with a single value, progress, which is a percentage value from 0-100.

Returns
The resource is returned on success.

Raises
ResourceTimeout if transition to delete failed to occur in the specified seconds.

Service Operations

class openstack.clustering.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

services(***query*)

Get a generator of services.

Returns
A generator of objects that are of type `Service`

Compute API

For details on how to use compute, see *Using OpenStack Compute*

The Compute Class

The compute high-level interface is available through the `compute` member of a *Connection* object. The `compute` member will only be added if the service is detected.

Server Operations

class openstack.compute.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_server(***attrs*)

Create a new server from attributes

Parameters

attrs (*dict*) Keyword arguments which will be used to create a *Server*, comprised of the properties on the Server class.

Returns

The results of server creation

Return type

Server

delete_server(*server*, *ignore_missing=True*, *force=False*)

Delete a server

Parameters

- **server** The value can be either the ID of a server or a *Server* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the server does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent server
- **force** (*bool*) When set to `True`, the server deletion will be forced immediately.

Returns

None

find_server(*name_or_id*, *ignore_missing=True*, ***, *details=True*, *all_projects=False*)

Find a single server

Parameters

- **name_or_id** The name or ID of a server.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

- **details** (*bool*) When set to `False` instances with only basic data will be returned. The default, `True`, will cause instances with full data to be returned.
- **all_projects** (*bool*) When set to `True`, search for server by name across all projects. Note that this will likely result in a higher chance of duplicates. Admin-only by default.

Returns
One *Server* or None

Raises
NotFoundException when no resource can be found.

Raises
DuplicateResource when multiple resources are found.

get_server(*server*)

Get a single server

Parameters
server The value can be the ID of a server or a *Server* instance.

Returns
One *Server*

Raises
NotFoundException when no resource can be found.

servers(*details=True*, *all_projects=False*, ***query*)

Retrieve a generator of servers

Parameters

- **details** (*bool*) When set to `False` instances with only basic data will be returned. The default, `True`, will cause instances with full data to be returned.
- **all_projects** (*bool*) When set to `True`, lists servers from all projects. Admin-only by default.
- **query** (*kwargs*) Optional query parameters to be sent to limit the servers being returned. Available parameters can be seen under https://docs.openstack.org/api-ref/compute/#list-servers

Returns
A generator of server instances.

update_server(*server*, ***attrs*)

Update a server

Parameters

- **server** Either the ID of a server or a *Server* instance.
- **attrs** The attributes to update on the server represented by `server`.

Returns
The updated server

Return type
Server

create_server_image(*server*, *name*, *metadata=None*, *wait=False*, *timeout=120*)

Create an image from a server

Parameters

- **server** Either the ID of a server or a *Server* instance.
- **name** (*str*) The name of the image to be created.
- **metadata** (*dict*) A dictionary of metadata to be set on the image.

Returns
Image object.

backup_server(*server*, *name*, *backup_type*, *rotation*)

Backup a server

Parameters

- **server** Either the ID of a server or a *Server* instance.
- **name** The name of the backup image.
- **backup_type** The type of the backup, for example, daily.
- **rotation** The rotation of the back up image, the oldest image will be removed when image count exceed the rotation count.

Returns
None

get_server_metadata(*server*)

Return a dictionary of metadata for a server

Parameters
server Either the ID of a server or a *Server* or ServerDetail instance.

Returns
A *Server* with the servers metadata. All keys and values are Unicode text.

Return type
Server

set_server_metadata(*server*, ***metadata*)

Update metadata for a server

Parameters

- **server** Either the ID of a server or a *Server* instance.
- **metadata** (*kwargs*) Key/value pairs to be updated in the servers metadata. No other metadata is modified by this call. All keys and values are stored as Unicode.

Returns
A *Server* with only the servers metadata. All keys and values are Unicode text.

Return type
Server

delete_server_metadata(*server*, *keys=None*)

Delete metadata for a server

Note: This method will do a HTTP DELETE request for every key in keys.

Parameters

- **server** Either the ID of a server or a *Server* instance.
- **keys** (*list*) The keys to delete. If left empty complete metadata will be removed.

Return type
None

wait_for_server(*server*, *status='ACTIVE'*, *failures=None*, *interval=2*, *wait=120*, *callback=None*)

Wait for a server to be in a particular status.

Parameters

- **server** (*Server*:) The *Server* to wait on to reach the specified status.
- **status** (*str*) Desired status.
- **failures** (list) Statuses that would be interpreted as failures.
- **interval** (*int*) Number of seconds to wait before to consecutive checks. Default to 2.
- **wait** Maximum number of seconds to wait before the change. Default to 120.
- **callback** (*callable*) A callback function. This will be called with a single value, progress, which is a percentage value from 0-100.

Returns
The resource is returned on success.

Raises
ResourceTimeout if transition to the desired status failed to occur in specified seconds.

Raises
ResourceFailure if the resource has transited to one of the failure statuses.

Raises
AttributeError if the resource does not have a status attribute.

Network Actions

class openstack.compute.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

`fetch_server_security_groups`(*server*)

Fetch security groups with details for a server.

Parameters

server Either the ID of a server or a *Server* instance.

Returns

updated *Server* instance

`add_security_group_to_server`(*server*, *security_group*)

Add a security group to a server

Parameters

- **server** Either the ID of a server or a *Server* instance.
- **security_group** Either the ID or name of a security group or a *SecurityGroup* instance.

Returns

None

`remove_security_group_from_server`(*server*, *security_group*)

Remove a security group from a server

Parameters

- **server** Either the ID of a server or a *Server* instance.
- **security_group** Either the ID or name of a security group or a *SecurityGroup* instance.

Returns

None

`add_fixed_ip_to_server`(*server*, *network_id*)

Adds a fixed IP address to a server instance.

Parameters

- **server** Either the ID of a server or a *Server* instance.
- **network_id** The ID of the network from which a fixed IP address is about to be allocated.

Returns

None

`remove_fixed_ip_from_server`(*server*, *address*)

Removes a fixed IP address from a server instance.

Parameters

- **server** Either the ID of a server or a *Server* instance.
- **address** The fixed IP address to be disassociated from the server.

Returns

None

add_floating_ip_to_server(*server*, *address*, *fixed_address=None*)

Adds a floating IP address to a server instance.

Parameters

- **server** Either the ID of a server or a *Server* instance.
- **address** The floating IP address to be added to the server.
- **fixed_address** The fixed IP address to be associated with the floating IP address. Used when the server is connected to multiple networks.

Returns
None

remove_floating_ip_from_server(*server*, *address*)

Removes a floating IP address from a server instance.

Parameters

- **server** Either the ID of a server or a *Server* instance.
- **address** The floating IP address to be disassociated from the server.

Returns
None

Starting, Stopping, etc.

class openstack.compute.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

reboot_server(*server*, *reboot_type*)

Reboot a server

Parameters

- **server** Either the ID of a server or a *Server* instance.
- **reboot_type** (*str*) The type of reboot to perform. HARD and SOFT are the current options.

Returns
None

pause_server(*server*)

Pauses a server and changes its status to PAUSED.

Parameters
server Either the ID of a server or a *Server* instance.

Returns
None

unpause_server(*server*)

Unpauses a paused server and changes its status to ACTIVE.

Parameters
server Either the ID of a server or a *Server* instance.

Returns
None

suspend_server(*server*)

Suspends a server and changes its status to SUSPENDED.

Parameters
server Either the ID of a server or a *Server* instance.

Returns
None

resume_server(*server*)

Resumes a suspended server and changes its status to ACTIVE.

Parameters
server Either the ID of a server or a *Server* instance.

Returns
None

lock_server(*server*, *locked_reason=None*)

Locks a server.

Parameters

- **server** Either the ID of a server or a *Server* instance.
- **locked_reason** The reason behind locking the server. Limited to 255 characters in length.

Returns
None

unlock_server(*server*)

Unlocks a locked server.

Parameters
server Either the ID of a server or a *Server* instance.

Returns
None

rescue_server(*server*, *admin_pass=None*, *image_ref=None*)

Puts a server in rescue mode and changes it status to RESCUE.

Parameters

- **server** Either the ID of a server or a *Server* instance.
- **admin_pass** The password for the rescued server. If you omit this parameter, the operation generates a new password.
- **image_ref** The image reference to use to rescue your server. This can be the image ID or its full URL. If you omit this parameter, the base image reference will be used.

Returns
None

`unrescue_server`(*server*)

Unrescues a server and changes its status to `ACTIVE`.

Parameters
server Either the ID of a server or a *Server* instance.

Returns
None

`evacuate_server`(*server*, *host=None*, *admin_pass=None*, *force=None*, ***, *on_shared_storage=None*)

Evacuates a server from a failed host to a new host.

Parameters

- **server** Either the ID of a server or a *Server* instance.
- **host** An optional parameter specifying the name or ID of the host to which the server is evacuated.
- **admin_pass** An optional parameter specifying the administrative password to access the evacuated or rebuilt server.
- **force** Force an evacuation by not verifying the provided destination host by the scheduler. (New in API version 2.29).
- **on_shared_storage** Whether the host is using shared storage. (Optional) (Only supported before API version 2.14)

Returns
None

`start_server`(*server*)

Starts a stopped server and changes its state to `ACTIVE`.

Parameters
server Either the ID of a server or a *Server* instance.

Returns
None

`stop_server`(*server*)

Stops a running server and changes its state to `SHUTOFF`.

Parameters
server Either the ID of a server or a *Server* instance.

Returns
None

`restore_server`(*server*)

Restore a soft-deleted server.

Parameters
server Either the ID of a server or a *Server* instance.

Returns
None

shelve_server(*server*)

Shelves a server.

All associated data and resources are kept but anything still in memory is not retained. Policy defaults enable only users with administrative role or the owner of the server to perform this operation. Cloud provides could change this permission though.

Parameters
server Either the ID of a server or a *Server* instance.

Returns
None

unshelve_server(*server*, *, *host=None*)

Unshelves or restores a shelved server.

Policy defaults enable only users with administrative role or the owner of the server to perform this operation. Cloud provides could change this permission though.

Parameters

- **server** Either the ID of a server or a *Server* instance.
- **host** An optional parameter specifying the name the compute host to unshelve to. (New in API version 2.91).

Returns
None

migrate_server(*server*, *, *host=None*)

Migrate a server from one host to another

Parameters

- **server** Either the ID of a server or a *Server* instance.
- **host** (*str*) The host to which to migrate the server.

Returns
None

live_migrate_server(*server*, *host=None*, *force=False*, *block_migration=None*, *disk_over_commit=None*)

Live migrate a server from one host to target host

Parameters

- **server** Either the ID of a server or a *Server* instance.
- **host** (*str*) The host to which to migrate the server. If the Nova service is too old, the host parameter implies force=True which causes the Nova scheduler to be bypassed. On such clouds, a `ValueError` will be thrown if `host` is given without `force`.
- **force** (*bool*) Force a live-migration by not verifying the provided destination host by the scheduler. This is unsafe and not recommended.

- **block_migration** Perform a block live migration to the destination host by the scheduler. Can be auto, True or False. Some clouds are too old to support auto, in which case a ValueError will be thrown. If omitted, the value will be auto on clouds that support it, and False on clouds that do not.
- **disk_over_commit** Whether to allow disk over-commit on the destination host. (Optional)

Returns
None

`get_server_console_output`(*server*, *length=None*)

Return the console output for a server.

Parameters

- **server** Either the ID of a server or a *Server* instance.
- **length** Optional number of line to fetch from the end of console log. All lines will be returned if this is not specified.

Returns
The console output as a dict. Control characters will be escaped to create a valid JSON string.

Modifying a Server

`class openstack.compute.v2._proxy.Proxy`(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

`change_server_password`(*server*, *new_password*)

Change the administrator password

Parameters

- **server** Either the ID of a server or a *Server* instance.
- **new_password** (*str*) The new password to be set.

Returns
None

`get_server_password`(*server*)

Get the administrator password

Parameters
server Either the ID of a server or a *Server* instance.

Returns
encrypted password.

`reset_server_state`(*server*, *state*)

Reset the state of server

Parameters

- **server** The server can be either the ID of a server or a *Server*.
- **state** The state of the server to be set, *active* or *error* are valid.

Returns
None

rebuild_server(*server*, *image*, ***attrs*)

Rebuild a server

Parameters

- **server** Either the ID of a server or a *Server* instance.
- **name** (*str*) The name of the server
- **admin_password** (*str*) The administrator password
- **preserve_ephemeral** (*bool*) Indicates whether the server is rebuilt with the preservation of the ephemeral partition. *Default: False*
- **image** (*str*) The id of an image to rebuild with. *Default: None*
- **access_ipv4** (*str*) The IPv4 address to rebuild with. *Default: None*
- **access_ipv6** (*str*) The IPv6 address to rebuild with. *Default: None*
- **metadata** (*dict*) A dictionary of metadata to rebuild with. *Default: None*
- **personality** A list of dictionaries, each including a **path** and **contents** key, to be injected into the rebuilt server at launch. *Default: None*

Returns
The rebuilt *Server* instance.

resize_server(*server*, *flavor*)

Resize a server

Parameters

- **server** Either the ID of a server or a *Server* instance.
- **flavor** Either the ID of a flavor or a *Flavor* instance.

Returns
None

confirm_server_resize(*server*)

Confirm a server resize

Parameters
server Either the ID of a server or a *Server* instance.

Returns
None

revert_server_resize(*server*)

Revert a server resize

Parameters
server Either the ID of a server or a *Server* instance.

Returns
None

Image Operations

class openstack.compute.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

delete_image(*image*, *ignore_missing=True*)

Delete an image

Parameters

- **image** The value can be either the ID of an image or a *Image* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the image does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent image.

Returns
`None`

find_image(*name_or_id*, *ignore_missing=True*)

Find a single image

Parameters

- **name_or_id** The name or ID of a image.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns
One *Image* or None

Raises
NotFoundException when no resource can be found.

Raises
DuplicateResource when multiple resources are found.

get_image(*image*)

Get a single image

Parameters
image The value can be the ID of an image or a *Image* instance.

Returns
One *Image*

Raises
NotFoundException when no resource can be found.

images(*details=True*, ***query*)

Return a generator of images

Parameters

- **details** (*bool*) When `True`, returns *Image* objects with all available properties, otherwise only basic properties are returned. *Default: "True"*
- **query** (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns

A generator of image objects

get_image_metadata(*image*)

Return a dictionary of metadata for an image

Parameters

image Either the ID of an image or a *Image* instance.

Returns

A *Image* with only the images metadata. All keys and values are Unicode text.

Return type

Image

set_image_metadata(*image*, ***metadata*)

Update metadata for an image

Parameters

- **image** Either the ID of an image or a *Image* instance.
- **metadata** (*kwargs*) Key/value pairs to be updated in the images metadata. No other metadata is modified by this call. All keys and values are stored as Unicode.

Returns

A *Image* with only the images metadata. All keys and values are Unicode text.

Return type

Image

delete_image_metadata(*image*, *keys=None*)

Delete metadata for an image

Note: This method will do a HTTP DELETE request for every key in keys.

Parameters

- **image** Either the ID of an image or a *Image* instance.
- **keys** (*list*) The keys to delete. If left empty complete metadata will be removed.

Return type

None

Flavor Operations

class openstack.compute.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

find_flavor(*name_or_id*, *ignore_missing=True*, ***, *get_extra_specs=False*, ***query*)

Find a single flavor

Parameters

- **name_or_id** The name or ID of a flavor.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **get_extra_specs** (*bool*) When set to `True` and extra_specs not present in the response will invoke additional API call to fetch extra_specs.
- **query** (*kwargs*) Optional query parameters to be sent to limit the flavors being returned.

Returns

One *Flavor* or None

Raises

NotFoundException when no resource can be found.

Raises

DuplicateResource when multiple resources are found.

create_flavor(***attrs*)

Create a new flavor from attributes

Parameters

attrs (*dict*) Keyword arguments which will be used to create a *Flavor*, comprised of the properties on the Flavor class.

Returns

The results of flavor creation

Return type

Flavor

delete_flavor(*flavor*, *ignore_missing=True*)

Delete a flavor

Parameters

- **flavor** The value can be either the ID of a flavor or a *Flavor* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the flavor does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent flavor.

Returns

None

get_flavor(*flavor*, *get_extra_specs=False*)

Get a single flavor

Parameters

- **flavor** The value can be the ID of a flavor or a *Flavor* instance.
- **get_extra_specs** (*bool*) When set to `True` and extra_specs not present in the response will invoke additional API call to fetch extra_specs.

Returns

One *Flavor*

Raises

NotFoundException when no resource can be found.

flavors(*details=True*, *get_extra_specs=False*, ***query*)

Return a generator of flavors

Parameters

- **details** (*bool*) When `True`, returns *Flavor* objects, with additional attributes filled.
- **get_extra_specs** (*bool*) When set to `True` and extra_specs not present in the response will invoke additional API call to fetch extra_specs.
- **query** (*kwargs*) Optional query parameters to be sent to limit the flavors being returned.

Returns

A generator of flavor objects

flavor_add_tenant_access(*flavor*, *tenant*)

Adds tenant/project access to flavor.

Parameters

- **flavor** Either the ID of a flavor or a *Flavor* instance.
- **tenant** (*str*) The UUID of the tenant.

Returns

One *Flavor*

flavor_remove_tenant_access(*flavor*, *tenant*)

Removes tenant/project access to flavor.

Parameters

- **flavor** Either the ID of a flavor or a *Flavor* instance.
- **tenant** (*str*) The UUID of the tenant.

Returns

One *Flavor*

get_flavor_access(*flavor*)

Lists tenants who have access to private flavor

Parameters

flavor Either the ID of a flavor or a *Flavor* instance.

Returns
List of dicts with flavor_id and tenant_id attributes.

`fetch_flavor_extra_specs`(*flavor*)

Lists Extra Specs of a flavor

Parameters
flavor Either the ID of a flavor or a *Flavor* instance.

Returns
One *Flavor*

`create_flavor_extra_specs`(*flavor*, *extra_specs*)

Lists Extra Specs of a flavor

Parameters

- **flavor** Either the ID of a flavor or a *Flavor* instance.
- **extra_specs** (*dict*) dict of extra specs

Returns
One *Flavor*

`get_flavor_extra_specs_property`(*flavor*, *prop*)

Get specific Extra Spec property of a flavor

Parameters

- **flavor** Either the ID of a flavor or a *Flavor* instance.
- **prop** (*str*) Property name.

Returns
String value of the requested property.

`update_flavor_extra_specs_property`(*flavor*, *prop*, *val*)

Update specific Extra Spec property of a flavor

Parameters

- **flavor** Either the ID of a flavor or a *Flavor* instance.
- **prop** (*str*) Property name.
- **val** (*str*) Property value.

Returns
String value of the requested property.

`delete_flavor_extra_specs_property`(*flavor*, *prop*)

Delete specific Extra Spec property of a flavor

Parameters

- **flavor** Either the ID of a flavor or a *Flavor* instance.
- **prop** (*str*) Property name.

Returns
None

Service Operations

class openstack.compute.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

update_service_forced_down(*service*, *host=None*, *binary=None*, *forced=True*)

Update service forced_down information

Parameters

- **service** Either the ID of a service or a *Service* instance.
- **host** (*str*) The host where service runs.
- **binary** (*str*) The name of service.
- **forced** (*bool*) Whether or not this service was forced down manually by an administrator after the service was fenced.

Returns

Updated service instance

Return type

class: *~openstack.compute.v2.service.Service*

disable_service(*service*, *host=None*, *binary=None*, *disabled_reason=None*)

Disable a service

Parameters

- **service** Either the ID of a service or a *Service* instance.
- **host** (*str*) The host where service runs.
- **binary** (*str*) The name of service.
- **disabled_reason** (*str*) The reason of force down a service.

Returns

Updated service instance

Return type

class: *~openstack.compute.v2.service.Service*

enable_service(*service*, *host=None*, *binary=None*)

Enable a service

Parameters

- **service** Either the ID of a service or a *Service* instance.
- **host** (*str*) The host where service runs.
- **binary** (*str*) The name of service.

Returns

Updated service instance

Return type
class: *~openstack.compute.v2.service.Service*

services(***query*)

Return a generator of service

Params dict query
Query parameters

Returns
A generator of service

Return type
class: *~openstack.compute.v2.service.Service*

find_service(*name_or_id*, *ignore_missing=True*, ***query*)

Find a service from name or id to get the corresponding info

Parameters

- **name_or_id** The name or id of a service
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Additional attributes like host

Returns
One: class:*~openstack.compute.v2.service.Service* or None

Raises
NotFoundException when no resource can be found.

Raises
DuplicateResource when multiple resources are found.

delete_service(*service*, *ignore_missing=True*)

Delete a service

Parameters

- **service** The value can be either the ID of a service or a *Service* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the service does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent service.

Returns
`None`

update_service(*service*, ***attrs*)

Update a service

Parameters

- **service** Either the ID of a service or a *Service* instance.
- **attrs** The attributes to update on the service represented by `service`.

Returns
The updated service

Return type
Service

Volume Attachment Operations

class openstack.compute.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_volume_attachment(*server*, *volume=None*, ***attrs*)

Create a new volume attachment from attributes

Parameters

- **server** The value can be either the ID of a server or a *Server* instance that the volume is attached to.
- **volume** The value can be either the ID of a volume or a *Volume* instance.
- **attrs** (*dict*) Keyword arguments which will be used to create a *VolumeAttachment*, comprised of the properties on the VolumeAttachment class.

Returns
The results of volume attachment creation

Return type
VolumeAttachment

update_volume_attachment(*server*, *volume*, *volume_id=None*, ***attrs*)

Update a volume attachment

Note that the underlying API expects a volume ID, not a volume attachment ID. There is currently no way to update volume attachments by their own ID.

Parameters

- **server** The value can be either the ID of a server or a *Server* instance that the volume is attached to.
- **volume** The value can be either the ID of a volume or a *Volume* instance.
- **volume_id** The ID of a volume to swap to. If this is not specified, we will default to not swapping the volume.
- **attrs** The attributes to update on the volume attachment represented by `volume_attachment`.

Returns
`None`

delete_volume_attachment(*server*, *volume*, *ignore_missing=True*)

Delete a volume attachment

Note that the underlying API expects a volume ID, not a volume attachment ID. There is currently no way to delete volume attachments by their own ID.

Parameters

- **server** The value can be either the ID of a server or a *Server* instance that the volume is attached to.
- **volume** The value can be the ID of a volume or a *Volume* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the volume attachment does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent volume attachment.

Returns
`None`

get_volume_attachment(*server*, *volume*)

Get a single volume attachment

Note that the underlying API expects a volume ID, not a volume attachment ID. There is currently no way to retrieve volume attachments by their own ID.

Parameters

- **server** The value can be either the ID of a server or a *Server* instance that the volume is attached to.
- **volume** The value can be the ID of a volume or a *Volume* instance.

Returns
One *VolumeAttachment*

Raises
NotFoundException when no resource can be found.

volume_attachments(*server*, ***query*)

Return a generator of volume attachments

Parameters
server The server can be either the ID of a server or a *Server*.

Params dict query
Query parameters

Returns
A generator of VolumeAttachment objects

Return type
VolumeAttachment

Keypair Operations

class openstack.compute.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_keypair(***attrs*)

Create a new keypair from attributes

Parameters

attrs (*dict*) Keyword arguments which will be used to create a *Keypair*, comprised of the properties on the Keypair class.

Returns

The results of keypair creation

Return type

Keypair

delete_keypair(*keypair*, *ignore_missing=True*, *user_id=None*)

Delete a keypair

Parameters

- **keypair** The value can be either the ID of a keypair or a *Keypair* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the keypair does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent keypair.
- **user_id** (*str*) Optional user_id owning the keypair

Returns

`None`

get_keypair(*keypair*, *user_id=None*)

Get a single keypair

Parameters

- **keypair** The value can be the ID of a keypair or a *Keypair* instance.
- **user_id** (*str*) Optional user_id owning the keypair

Returns

One *Keypair*

Raises

NotFoundException when no resource can be found.

find_keypair(*name_or_id*, *ignore_missing=True*, ***, *user_id=None*)

Find a single keypair

Parameters

- **name_or_id** The name or ID of a keypair.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **user_id** (*str*) Optional user_id owning the keypair

Returns

One *Keypair* or None

Raises

NotFoundException when no resource can be found.

Raises
DuplicateResource when multiple resources are found.

keypairs(***query*)

Return a generator of keypairs

Parameters
query (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of keypair objects

Return type
Keypair

Server IPs

class openstack.compute.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

server_ips(*server*, *network_label=None*)

Return a generator of server IPs

Parameters

- **server** The server can be either the ID of a server or a *Server*.
- **network_label** The name of a particular network to list IP addresses from.

Returns
A generator of ServerIP objects

Return type
ServerIP

Server Group Operations

class openstack.compute.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_server_group(***attrs*)

Create a new server group from attributes

Parameters
attrs (*dict*) Keyword arguments which will be used to create a *ServerGroup*, comprised of the properties on the ServerGroup class.

Returns
The results of server group creation

Return type
ServerGroup

delete_server_group(*server_group*, *ignore_missing=True*)

Delete a server group

Parameters

- **server_group** The value can be either the ID of a server group or a *ServerGroup* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the server group does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent server group.

Returns
None

find_server_group(*name_or_id*, *ignore_missing=True*, ***, *all_projects=False*)

Find a single server group

Parameters

- **name_or_id** The name or ID of a server group.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **all_projects** (*bool*) When set to `True`, search for server groups by name across all projects. Note that this will likely result in a higher chance of duplicates. Admin-only by default.

Returns
One *ServerGroup* or None

Raises
NotFoundException when no resource can be found.

Raises
DuplicateResource when multiple resources are found.

get_server_group(*server_group*)

Get a single server group

Parameters
server_group The value can be the ID of a server group or a *ServerGroup* instance.

Returns
A *ServerGroup* object.

Raises
NotFoundException when no resource can be found.

server_groups(***, *all_projects=False*, ***query*)

Return a generator of server groups

Parameters

- **all_projects** (*bool*) When set to `True`, lists servers groups from all projects. Admin-only by default.
- **query** (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of ServerGroup objects

Return type
ServerGroup

Server Interface Operations

class openstack.compute.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_server_interface(*server*, ***attrs*)

Create a new server interface from attributes

Parameters

- **server** The server can be either the ID of a server or a *Server* instance that the interface belongs to.
- **attrs** (*dict*) Keyword arguments which will be used to create a *ServerInterface*, comprised of the properties on the ServerInterface class.

Returns
The results of server interface creation

Return type
ServerInterface

delete_server_interface(*server_interface*, *server=None*, *ignore_missing=True*)

Delete a server interface

Parameters

- **server_interface** The value can be either the ID of a server interface or a *ServerInterface* instance.
- **server** This parameter need to be specified when ServerInterface ID is given as value. It can be either the ID of a server or a *Server* instance that the interface belongs to.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the server interface does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent server interface.

Returns
`None`

get_server_interface(*server_interface*, *server=None*)

Get a single server interface

Parameters

- **server_interface** The value can be the ID of a server interface or a *ServerInterface* instance.
- **server** This parameter need to be specified when ServerInterface ID is given as value. It can be either the ID of a server or a *Server* instance that the interface belongs to.

Returns
One *ServerInterface*

Raises
NotFoundException when no resource can be found.

server_interfaces(*server*, ***query*)

Return a generator of server interfaces

Parameters

- **server** The server can be either the ID of a server or a *Server*.
- **query** Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of ServerInterface objects

Return type
ServerInterface

Server Tag Operations

class openstack.compute.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

add_tag_to_server(*server*, *tag*)

Add a tag to a server.

Parameters

- **server** Either the ID of a server or a *Server* instance.
- **tag** The tag to add.

Returns
None

remove_tag_from_server(*server*, *tag*)

Remove a tag from a server.

Parameters

- **server** Either the ID of a server or a *Server* instance.

- **tag** The tag to remove.

Returns
None

remove_tags_from_server(*server*)

Remove all tags from a server.

Parameters

- **server** Either the ID of a server or a *Server* instance.
- **tag** The tag to remove.

Returns
None

Availability Zone Operations

class openstack.compute.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

availability_zones(*details=False*)

Return a generator of availability zones

Parameters
details (*bool*) Return extra details about the availability zones. This defaults to *False* as it generally requires extra permission.

Returns
A generator of availability zone

Return type
AvailabilityZone

Limits Operations

class openstack.compute.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

get_limits(***query*)

Retrieve limits that are applied to the projects account

Returns
A Limits object, including both *AbsoluteLimits* and RateLimits

Return type
Limits

Hypervisor Operations

class openstack.compute.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

hypervisors(*details=False*, ***query*)

Return a generator of hypervisors

Parameters

- **details** (*bool*) When set to the default, `False`, *Hypervisor* instances will be returned with only basic information populated.
- **query** (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns

A generator of hypervisor

Return type

class: *~openstack.compute.v2.hypervisor.Hypervisor*

find_hypervisor(*name_or_id*, *ignore_missing=True*, ***, *details=True*)

Find a single hypervisor

Parameters

- **name_or_id** The name or ID of a hypervisor
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **details** (*bool*) When set to `False` instances with only basic data will be returned. The default, `True`, will cause instances with full data to be returned.

Returns

One: class:*~openstack.compute.v2.hypervisor.Hypervisor* or None

Raises

NotFoundException when no resource can be found.

Raises

DuplicateResource when multiple resources are found.

get_hypervisor(*hypervisor*)

Get a single hypervisor

Parameters

hypervisor The value can be the ID of a hypervisor or a *Hypervisor* instance.

Returns

A *Hypervisor* object.

Raises
NotFoundException when no resource can be found.

get_hypervisor_uptime(*hypervisor*)

Get uptime information for hypervisor

Parameters
hypervisor The value can be the ID of a hypervisor or a *Hypervisor* instance.

Returns
A *Hypervisor* object.

Raises
NotFoundException when no resource can be found.

Extension Operations

class openstack.compute.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

find_extension(*name_or_id*, *ignore_missing=True*)

Find a single extension

Parameters

- **name_or_id** The name or ID of an extension.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns
One *Extension* or None

Raises
NotFoundException when no resource can be found.

Raises
DuplicateResource when multiple resources are found.

extensions()

Retrieve a generator of extensions

Returns
A generator of extension instances.

Return type
Extension

QuotaClassSet Operations

class openstack.compute.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

get_quota_class_set(*quota_class_set='default'*)

Get a single quota class set

Only one quota class is permitted, `default`.

Parameters

quota_class_set The value can be the ID of a quota class set (only `default` is supported) or a `QuotaClassSet` instance.

Returns

One `QuotaClassSet`

Raises

NotFoundException when no resource can be found.

update_quota_class_set(*quota_class_set*, ***attrs*)

Update a QuotaClassSet.

Only one quota class is permitted, `default`.

Parameters

- **quota_class_set** Either the ID of a quota class set (only `default` is supported) or a `QuotaClassSet` instance.
- **attrs** The attributes to update on the QuotaClassSet represented by `quota_class_set`.

Returns

The updated QuotaSet

Return type

QuotaSet

QuotaSet Operations

class openstack.compute.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

get_quota_set(*project*, *usage=False*, ***query*)

Show QuotaSet information for the project.

Parameters

- **project** ID or instance of `Project` of the project for which the quota should be retrieved

- **usage** (*bool*) When set to `True` quota usage and reservations would be filled.
- **query** (*dict*) Additional query parameters to use.

Returns
One *QuotaSet*

Raises
NotFoundException when no resource can be found.

get_quota_set_defaults(*project*)

Show QuotaSet defaults for the project.

Parameters
project ID or instance of `Project` of the project for which the quota should be retrieved

Returns
One *QuotaSet*

Raises
NotFoundException when no resource can be found.

revert_quota_set(*project*, ***query*)

Reset Quota for the project/user.

Parameters

- **project** ID or instance of `Project` of the project for which the quota should be reset.
- **query** (*dict*) Additional parameters to be used.

Returns
`None`

update_quota_set(*project*, *, *user=None*, ***attrs*)

Update a QuotaSet.

Parameters

- **project** ID or instance of `Project` of the project for which the quota should be reset.
- **user_id** Optional ID of the user to set quotas as.
- **attrs** The attributes to update on the QuotaSet represented by `quota_set`.

Returns
The updated QuotaSet

Return type
QuotaSet

Server Migration Operations

class openstack.compute.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

abort_server_migration(*server_migration*, *server*, *ignore_missing=True*)

Abort an in-progress server migration

Parameters

- **server_migration** The value can be either the ID of a server migration or a *ServerMigration* instance.
- **server** This parameter needs to be specified when ServerMigration ID is given as value. It can be either the ID of a server or a *Server* instance that the migration belongs to.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the server migration does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent server migration.

Returns

`None`

force_complete_server_migration(*server_migration*, *server=None*)

Force complete an in-progress server migration

Parameters

- **server_migration** The value can be either the ID of a server migration or a *ServerMigration* instance.
- **server** This parameter needs to be specified when ServerMigration ID is given as value. It can be either the ID of a server or a *Server* instance that the migration belongs to.

Returns

`None`

get_server_migration(*server_migration*, *server*, *ignore_missing=True*)

Get a single server migration

Parameters

- **server_migration** The value can be the ID of a server migration or a *ServerMigration* instance.
- **server** This parameter need to be specified when ServerMigration ID is given as value. It can be either the ID of a server or a *Server* instance that the migration belongs to.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the server migration does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent server migration.

Returns

One *ServerMigration*

Raises
NotFoundException when no resource can be found.

server_migrations(*server*)

Return a generator of migrations for a server.

Parameters
server The server can be either the ID of a server or a *Server*.

Returns
A generator of ServerMigration objects

Return type
ServerMigration

Migration Operations

class openstack.compute.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

migrations(***query*)

Return a generator of migrations for all servers.

Parameters
query (*kwargs*) Optional query parameters to be sent to limit the migrations being returned.

Returns
A generator of Migration objects

Return type
Migration

Helpers

class openstack.compute.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

wait_for_delete(*res*, *interval=2*, *wait=120*, *callback=None*)

Wait for a resource to be deleted.

Parameters

- **res** The resource to wait on to be deleted.
- **interval** Number of seconds to wait before to consecutive checks.
- **wait** Maximum number of seconds to wait before the change.
- **callback** A callback function. This will be called with a single value, progress, which is a percentage value from 0-100.

Returns
The resource is returned on success.

Raises
ResourceTimeout if transition to delete failed to occur in the specified seconds.

Container Infrastructure Management

Cluster Operations

class openstack.container_infrastructure_management.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_cluster(***attrs*)

Create a new cluster from attributes

Parameters
attrs (*dict*) Keyword arguments which will be used to create a *Cluster*, comprised of the properties on the Cluster class.

Returns
The results of cluster creation

Return type
Cluster

delete_cluster(*cluster*, *ignore_missing=True*)

Delete a cluster

Parameters

- **cluster** The value can be either the ID of a cluster or a *Cluster* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the cluster does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent cluster.

Returns
`None`

find_cluster(*name_or_id*, *ignore_missing=True*)

Find a single cluster

Parameters

- **name_or_id** The name or ID of a cluster.

- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns
One *Cluster* or None

get_cluster(*cluster*)

Get a single cluster

Parameters
cluster The value can be the ID of a cluster or a *Cluster* instance.

Returns
One *Cluster*

Raises
NotFoundException when no resource can be found.

clusters(***query*)

Return a generator of clusters

Parameters
query (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of cluster objects

Return type
Cluster

update_cluster(*cluster*, ***attrs*)

Update a cluster

Parameters

- **cluster** Either the id of a cluster or a *Cluster* instance.
- **attrs** The attributes to update on the cluster represented by `cluster`.

Returns
The updated cluster

Return type
Cluster

Cluster Certificates Operations

class openstack.container_infrastructure_management.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_cluster_certificate(***attrs*)

Create a new cluster_certificate from CSR

Parameters

attrs (*dict*) Keyword arguments which will be used to create a *ClusterCertificate*, comprised of the properties on the ClusterCertificate class.

Returns

The results of cluster_certificate creation

Return type

ClusterCertificate

get_cluster_certificate(*cluster_certificate*)

Get a single cluster_certificate

Parameters

cluster_certificate The value can be the ID of a cluster_certificate or a *ClusterCertificate* instance.

Returns

One *ClusterCertificate*

Raises

NotFoundException when no resource can be found.

Cluster Templates Operations

class openstack.container_infrastructure_management.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_cluster_template(***attrs*)

Create a new cluster_template from attributes

Parameters
attrs (*dict*) Keyword arguments which will be used to create a *ClusterTemplate*, comprised of the properties on the ClusterTemplate class.

Returns
The results of cluster_template creation

Return type
ClusterTemplate

delete_cluster_template(*cluster_template*, *ignore_missing=True*)

Delete a cluster_template

Parameters

- **cluster_template** The value can be either the ID of a cluster_template or a *ClusterTemplate* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the cluster_template does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent cluster_template.

Returns
`None`

find_cluster_template(*name_or_id*, *ignore_missing=True*)

Find a single cluster_template

Parameters

- **name_or_id** The name or ID of a cluster_template.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns
One *ClusterTemplate* or None

get_cluster_template(*cluster_template*)

Get a single cluster_template

Parameters
cluster_template The value can be the ID of a cluster_template or a *ClusterTemplate* instance.

Returns
One *ClusterTemplate*

Raises
NotFoundException when no resource can be found.

cluster_templates(***query*)

Return a generator of cluster_templates

Parameters
query (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of cluster_template objects

Return type
ClusterTemplate

update_cluster_template(*cluster_template*, ***attrs*)

Update a cluster_template

Parameters

- **cluster_template** Either the id of a cluster_template or a *ClusterTemplate* instance.
- **attrs** The attributes to update on the cluster_template represented by `cluster_template`.

Returns
The updated cluster_template

Return type
ClusterTemplate

Service Operations

class openstack.container_infrastructure_management.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

services()

Return a generator of services

Returns
A generator of service objects

Return type
Service

Database API

For details on how to use database, see *Using OpenStack Database*

The Database Class

The database high-level interface is available through the `database` member of a *Connection* object. The `database` member will only be added if the service is detected.

Database Operations

class openstack.database.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_database(*instance*, ***attrs*)

Create a new database from attributes

Parameters

- **instance** This can be either the ID of an instance or a *Instance*
- **attrs** (*dict*) Keyword arguments which will be used to create a *Database*, comprised of the properties on the Database class.

Returns
The results of server creation

Return type
Database

delete_database(*database*, *instance=None*, *ignore_missing=True*)

Delete a database

Parameters

- **database** The value can be either the ID of a database or a *Database* instance.
- **instance** This parameter needs to be specified when an ID is given as *database*. It can be either the ID of an instance or a *Instance*
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the database does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent database.

Returns
`None`

find_database(*name_or_id*, *instance*, *ignore_missing=True*)

Find a single database

Parameters

- **name_or_id** The name or ID of a database.
- **instance** This can be either the ID of an instance or a *Instance*
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns
One *Database* or None

databases(*instance*, ***query*)

Return a generator of databases

Parameters

- **instance** This can be either the ID of an instance or a *Instance* instance that the interface belongs to.
- **query** (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of database objects

Return type
Database

get_database(*database*, *instance=None*)

Get a single database

Parameters

- **instance** This parameter needs to be specified when an ID is given as *database*. It can be either the ID of an instance or a *Instance*
- **database** The value can be the ID of a database or a *Database* instance.

Returns
One *Database*

Raises
NotFoundException when no resource can be found.

Flavor Operations

class openstack.database.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

find_flavor(*name_or_id*, *ignore_missing=True*)

Find a single flavor

Parameters

- **name_or_id** The name or ID of a flavor.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns
One *Flavor* or None

get_flavor(*flavor*)

Get a single flavor

Parameters
flavor The value can be the ID of a flavor or a *Flavor* instance.

Returns
One *Flavor*

Raises
NotFoundException when no resource can be found.

flavors(***query*)

Return a generator of flavors

Parameters
query (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of flavor objects

Return type
Flavor

Instance Operations

class openstack.database.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_instance(***attrs*)

Create a new instance from attributes

Parameters
attrs (*dict*) Keyword arguments which will be used to create a *Instance*, comprised of the properties on the Instance class.

Returns
The results of server creation

Return type
Instance

delete_instance(*instance*, *ignore_missing=True*)

Delete an instance

Parameters

- **instance** The value can be either the ID of an instance or a *Instance* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the instance does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent instance.

Returns
`None`

find_instance(*name_or_id*, *ignore_missing=True*)

Find a single instance

Parameters

- **name_or_id** The name or ID of a instance.

- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns
One *Instance* or None

get_instance(*instance*)

Get a single instance

Parameters
instance The value can be the ID of an instance or a *Instance* instance.

Returns
One *Instance*

Raises
NotFoundException when no resource can be found.

instances(***query*)

Return a generator of instances

Parameters
query (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of instance objects

Return type
Instance

update_instance(*instance*, ***attrs*)

Update a instance

Parameters

- **instance** Either the id of a instance or a *Instance* instance.
- **attrs** The attributes to update on the instance represented by `instance`.

Returns
The updated instance

Return type
Instance

User Operations

class openstack.database.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_user(*instance*, ***attrs*)

Create a new user from attributes

Parameters

- **instance** This can be either the ID of an instance or a *Instance*
- **attrs** (*dict*) Keyword arguments which will be used to create a *User*, comprised of the properties on the User class.

Returns
The results of server creation

Return type
User

delete_user(*user*, *instance=None*, *ignore_missing=True*)

Delete a user

Parameters

- **user** The value can be either the ID of a user or a *User* instance.
- **instance** This parameter needs to be specified when an ID is given as *user*. It can be either the ID of an instance or a *Instance*
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the user does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent user.

Returns
`None`

find_user(*name_or_id*, *instance*, *ignore_missing=True*)

Find a single user

Parameters

- **name_or_id** The name or ID of a user.
- **instance** This can be either the ID of an instance or a *Instance*
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns
One *User* or None

users(*instance*, ***query*)

Return a generator of users

Parameters

- **instance** This can be either the ID of an instance or a *Instance*
- **query** (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of user objects

Return type
User

get_user(*user*, *instance=None*)

Get a single user

Parameters

- **user** The value can be the ID of a user or a *User* instance.
- **instance** This parameter needs to be specified when an ID is given as *database*. It can be either the ID of an instance or a *Instance*

Returns

One *User*

Raises

NotFoundException when no resource can be found.

DNS API

For details on how to use dns, see *Using OpenStack DNS*

The DNS Class

The dns high-level interface is available through the `dns` member of a *Connection* object. The `dns` member will only be added if the service is detected.

DNS Zone Operations

class openstack.dns.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

zones(***query*)

Retrieve a generator of zones

Parameters

query (*dict*) Optional query parameters to be sent to limit the resources being returned.

- *name*: Zone Name field.
- *type*: Zone Type field.
- *email*: Zone email field.
- *status*: Status of the zone.
- *ttl*: TTL field filter.abs
- *description*: Zone description field filter.

Returns

A generator of zone *Zone* instances.

create_zone(***attrs*)

Create a new zone from attributes

Parameters
attrs (*dict*) Keyword arguments which will be used to create a *Zone*, comprised of the properties on the Zone class.

Returns
The results of zone creation.

Return type
Zone

get_zone(*zone*)

Get a zone

Parameters
zone The value can be the ID of a zone or a *Zone* instance.

Returns
Zone instance.

Return type
Zone

delete_zone(*zone*, *ignore_missing=True*, *delete_shares=False*)

Delete a zone

Parameters

- **zone** The value can be the ID of a zone or a *Zone* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the zone does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent zone.
- **delete_shares** (*bool*) When True, delete the zone shares along with the zone.

Returns
Zone been deleted

Return type
Zone

find_zone(*name_or_id*, *ignore_missing=True*)

Find a single zone

Parameters

- **name_or_id** The name or ID of a zone
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the zone does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent zone.

Returns
Zone

abandon_zone(*zone*, ***attrs*)

Abandon Zone

Parameters
zone The value can be the ID of a zone to be abandoned or a *ZoneExport* instance.

Returns
None

xfr_zone(*zone*, ***attrs*)

Trigger update of secondary Zone

Parameters
zone The value can be the ID of a zone to be abandoned or a *ZoneExport* instance.

Returns
None

Recordset Operations

class openstack.dns.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

recordsets(*zone=None*, ***query*)

Retrieve a generator of recordsets

Parameters

- **zone** The optional value can be the ID of a zone or a *Zone* instance. If it is not given all recordsets for all zones of the tenant would be retrieved
- **query** (*dict*) Optional query parameters to be sent to limit the resources being returned.
  - *name*: Recordset Name field.
  - *type*: Type field.
  - *status*: Status of the recordset.
  - *ttl*: TTL field filter.
  - *description*: Recordset description field filter.

Returns
A generator of zone (*Recordset*) instances

create_recordset(*zone*, ***attrs*)

Create a new recordset in the zone

Parameters

- **zone** The value can be the ID of a zone or a *Zone* instance.
- **attrs** (*dict*) Keyword arguments which will be used to create a *Recordset*, comprised of the properties on the Recordset class.

Returns
The results of zone creation

Return type
Recordset

update_recordset(*recordset*, ***attrs*)

Update Recordset attributes

Parameters
attrs (*dict*) Keyword arguments which will be used to create a *Recordset*, comprised of the properties on the Recordset class.

Returns
The results of zone creation

Return type
Recordset

get_recordset(*recordset*, *zone*)

Get a recordset

Parameters

- **zone** The value can be the ID of a zone or a *Zone* instance.
- **recordset** The value can be the ID of a recordset or a *Recordset* instance.

Returns
Recordset instance

Return type
Recordset

delete_recordset(*recordset*, *zone=None*, *ignore_missing=True*)

Delete a zone

Parameters

- **recordset** The value can be the ID of a recordset or a *Recordset* instance.
- **zone** The value can be the ID of a zone or a *Zone* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the zone does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent zone.

Returns
Recordset instance been deleted

Return type
Recordset

Zone Import Operations

class openstack.dns.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

zone_imports(***query*)

Retrieve a generator of zone imports

Parameters

query (*dict*) Optional query parameters to be sent to limit the resources being returned.

- *zone_id*: Zone I field.
- *message*: Message field.
- *status*: Status of the zone import record.

Returns

A generator of zone *ZoneImport* instances.

create_zone_import(***attrs*)

Create a new zone import from attributes

Parameters

attrs (*dict*) Keyword arguments which will be used to create a *ZoneImport*, comprised of the properties on the ZoneImport class.

Returns

The results of zone creation.

Return type

ZoneImport

get_zone_import(*zone_import*)

Get a zone import record

Parameters

zone The value can be the ID of a zone import or a *ZoneImport* instance.

Returns

ZoneImport instance.

Return type

ZoneImport

delete_zone_import(*zone_import*, *ignore_missing=True*)

Delete a zone import

Parameters

- **zone_import** The value can be the ID of a zone import or a *ZoneImport* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the zone does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent zone.

Returns

None

Zone Export Operations

class openstack.dns.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

zone_exports(***query*)

Retrieve a generator of zone exports

Parameters

query (*dict*) Optional query parameters to be sent to limit the resources being returned.

- *zone_id*: Zone I field.
- *message*: Message field.
- *status*: Status of the zone import record.

Returns

A generator of zone *ZoneExport* instances.

create_zone_export(*zone*, ***attrs*)

Create a new zone export from attributes

Parameters

- **zone** The value can be the ID of a zone to be exported or a *ZoneExport* instance.
- **attrs** (*dict*) Keyword arguments which will be used to create a *ZoneExport*, comprised of the properties on the ZoneExport class.

Returns

The results of zone creation.

Return type

ZoneExport

get_zone_export(*zone_export*)

Get a zone export record

Parameters

zone The value can be the ID of a zone import or a *ZoneExport* instance.

Returns

ZoneExport instance.

Return type

ZoneExport

get_zone_export_text(*zone_export*)

Get a zone export record as text

Parameters

zone The value can be the ID of a zone import or a *ZoneExport* instance.

Returns

ZoneExport instance.

Return type
ZoneExport

`delete_zone_export`(*zone_export*, *ignore_missing=True*)

Delete a zone export

Parameters

- **zone_export** The value can be the ID of a zone import or a *ZoneExport* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the zone does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent zone.

Returns
None

FloatingIP Operations

`class openstack.dns.v2._proxy.Proxy`(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

`floating_ips`(***query*)

Retrieve a generator of recordsets

Parameters
query (*dict*) Optional query parameters to be sent to limit the resources being returned.

- *name*: Recordset Name field.
- *type*: Type field.
- *status*: Status of the recordset.
- *ttl*: TTL field filter.
- *description*: Recordset description field filter.

Returns
A generator of floatingips (*FloatingIP*) instances

`get_floating_ip`(*floating_ip*)

Get a Floating IP

Parameters
floating_ip The value can be the ID of a floating ip or a *FloatingIP* instance. The ID is in format region_name:floatingip_id

Returns
FloatingIP instance.

Return type
FloatingIP

update_floating_ip(*floating_ip*, ***attrs*)

Update floating ip attributes

Parameters

- **floating_ip** The id or an instance of FloatingIP.
- **attrs** (*dict*) attributes for update on FloatingIP.

Return type
FloatingIP

Zone Transfer Operations

class openstack.dns.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

zone_transfer_requests(***query*)

Retrieve a generator of zone transfer requests

Parameters
query (*dict*) Optional query parameters to be sent to limit the resources being returned.

- *status*: Status of the recordset.

Returns
A generator of transfer requests (*ZoneTransferRequest*) instances

get_zone_transfer_request(*request*)

Get a ZoneTransfer Request info

Parameters
request The value can be the ID of a transfer request or a *ZoneTransferRequest* instance.

Returns
Zone transfer request instance.

Return type
ZoneTransferRequest

create_zone_transfer_request(*zone*, ***attrs*)

Create a new ZoneTransfer Request from attributes

Parameters

- **zone** The value can be the ID of a zone to be transferred or a *ZoneExport* instance.
- **attrs** (*dict*) Keyword arguments which will be used to create a *ZoneTransferRequest*, comprised of the properties on the ZoneTransferRequest class.

Returns
The results of zone transfer request creation.

Return type
ZoneTransferRequest

`update_zone_transfer_request`(*request*, ***attrs*)

Update ZoneTransfer Request attributes

Parameters

- **floating_ip** The id or an instance of *ZoneTransferRequest*.
- **attrs** (*dict*) attributes for update on *ZoneTransferRequest*.

Return type
ZoneTransferRequest

`delete_zone_transfer_request`(*request*, *ignore_missing=True*)

Delete a ZoneTransfer Request

Parameters

- **request** The value can be the ID of a zone transfer request or a *ZoneTransferRequest* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the zone does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent zone.

Returns
None

`zone_transfer_accepts`(***query*)

Retrieve a generator of zone transfer accepts

Parameters
query (*dict*) Optional query parameters to be sent to limit the resources being returned.

- *status*: Status of the recordset.

Returns
A generator of transfer accepts (*ZoneTransferAccept*) instances

`get_zone_transfer_accept`(*accept*)

Get a ZoneTransfer Accept info

Parameters
request The value can be the ID of a transfer accept or a *ZoneTransferAccept* instance.

Returns
Zone transfer request instance.

Return type
ZoneTransferAccept

`create_zone_transfer_accept`(***attrs*)

Create a new ZoneTransfer Accept from attributes

Parameters
attrs (*dict*) Keyword arguments which will be used to create a

ZoneTransferAccept, comprised of the properties on the ZoneTransferAccept class.

Returns
The results of zone transfer request creation.

Return type
ZoneTransferAccept

Zone Share Operations

class openstack.dns.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

zone_shares(*zone*, ***query*)

Retrieve a generator of zone shares

Parameters

- **zone** The zone ID or a *Zone* instance
- **query** (*dict*) Optional query parameters to be sent to limit the resources being returned.
  - *target_project_id*: The target project ID field.

Returns
A generator of zone shares *ZoneShare* instances.

get_zone_share(*zone*, *zone_share*)

Get a zone share

Parameters

- **zone** The value can be the ID of a zone or a *Zone* instance.
- **zone_share** The zone_share can be either the ID of the zone share or a *ZoneShare* instance that the zone share belongs to.

Returns
ZoneShare instance.

Return type
ZoneShare

find_zone_share(*zone*, *zone_share_id*, *ignore_missing=True*)

Find a single zone share

Parameters

- **zone** The value can be the ID of a zone or a *Zone* instance.
- **zone_share_id** The zone share ID
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the zone share does not exist. When set to `True`, None will be returned when attempting to find a nonexistent zone share.

Returns
ZoneShare

`create_zone_share`(*zone*, ***attrs*)

Create a new zone share from attributes

Parameters

- **zone** The zone ID or a *Zone* instance
- **attrs** (*dict*) Keyword arguments which will be used to create a *ZoneShare*, comprised of the properties on the ZoneShare class.

Returns
The results of zone share creation

Return type
ZoneShare

`delete_zone_share`(*zone*, *zone_share*, *ignore_missing=True*)

Delete a zone share

Parameters

- **zone** The zone ID or a *Zone* instance
- **zone_share** The zone_share can be either the ID of the zone share or a *ZoneShare* instance that the zone share belongs to.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the zone share does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent zone share.

Returns
None

Limit Operations

`class openstack.dns.v2._proxy.Proxy`(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

`limits`(***query*)

Retrieve a generator of limits

Returns
A generator of limits (*Limit*) instances

Service Status Operations

`class openstack.dns.v2._proxy.Proxy`(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

service_statuses()

Retrieve a generator of service statuses

Returns
A generator of service statuses *ServiceStatus* instances.

get_service_status(*service*)

Get a status of a service in the Designate system

Parameters
service The value can be the ID of a service or a *ServiceStatus* instance.

Returns
ServiceStatus instance.

Return type
ServiceStatus

Identity API v2

For details on how to use identity, see *Using OpenStack Identity*

The Identity v2 Class

The identity high-level interface is available through the `identity` member of a *Connection* object. The `identity` member will only be added if the service is detected.

Extension Operations

class openstack.identity.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

extensions()

Retrieve a generator of extensions

Returns
A generator of extension instances.

Return type
Extension

get_extension(*extension*)

Get a single extension

Parameters
extension The value can be the ID of an extension or a *Extension* instance.

Returns
One *Extension*

Raises
NotFoundException when no extension can be found.

User Operations

class openstack.identity.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_user(***attrs*)

Create a new user from attributes

Parameters

attrs (*dict*) Keyword arguments which will be used to create a *User*, comprised of the properties on the User class.

Returns

The results of user creation

Return type

User

delete_user(*user*, *ignore_missing=True*)

Delete a user

Parameters

- **user** The value can be either the ID of a user or a *User* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the user does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent user.

Returns

`None`

find_user(*name_or_id*, *ignore_missing=True*)

Find a single user

Parameters

- **name_or_id** The name or ID of a user.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns

One *User* or None

get_user(*user*)

Get a single user

Parameters

user The value can be the ID of a user or a *User* instance.

Returns

One *User*

Raises

NotFoundException when no resource can be found.

users(***query*)

Retrieve a generator of users

Parameters
query (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of user instances.

Return type
User

update_user(*user*, ***attrs*)

Update a user

Parameters

- **user** Either the ID of a user or a *User* instance.
- **attrs** The attributes to update on the user represented by `user`.

Returns
The updated user

Return type
User

Role Operations

class openstack.identity.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_role(***attrs*)

Create a new role from attributes

Parameters
attrs (*dict*) Keyword arguments which will be used to create a *Role*, comprised of the properties on the Role class.

Returns
The results of role creation

Return type
Role

delete_role(*role*, *ignore_missing=True*)

Delete a role

Parameters

- **role** The value can be either the ID of a role or a *Role* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the role does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent role.

Returns
None

find_role(*name_or_id*, *ignore_missing=True*)

Find a single role

Parameters

- **name_or_id** The name or ID of a role.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns
One *Role* or None

get_role(*role*)

Get a single role

Parameters
role The value can be the ID of a role or a *Role* instance.

Returns
One *Role*

Raises
NotFoundException when no resource can be found.

roles(***query*)

Retrieve a generator of roles

Parameters
query (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of role instances.

Return type
Role

update_role(*role*, ***attrs*)

Update a role

Parameters

- **role** Either the ID of a role or a *Role* instance.
- **attrs** The attributes to update on the role represented by `role`.

Returns
The updated role

Return type
Role

Tenant Operations

class openstack.identity.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_tenant(***attrs*)

Create a new tenant from attributes

Parameters

attrs (*dict*) Keyword arguments which will be used to create a *Tenant*, comprised of the properties on the Tenant class.

Returns

The results of tenant creation

Return type

Tenant

delete_tenant(*tenant*, *ignore_missing=True*)

Delete a tenant

Parameters

- **tenant** The value can be either the ID of a tenant or a *Tenant* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the tenant does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent tenant.

Returns

`None`

find_tenant(*name_or_id*, *ignore_missing=True*)

Find a single tenant

Parameters

- **name_or_id** The name or ID of a tenant.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns

One *Tenant* or None

get_tenant(*tenant*)

Get a single tenant

Parameters

tenant The value can be the ID of a tenant or a *Tenant* instance.

Returns

One *Tenant*

Raises

NotFoundException when no resource can be found.

tenants(***query*)

Retrieve a generator of tenants

Parameters

query (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns

A generator of tenant instances.

Return type

Tenant

update_tenant(*tenant*, ***attrs*)

Update a tenant

Parameters

- **tenant** Either the ID of a tenant or a *Tenant* instance.
- **attrs** The attributes to update on the tenant represented by `tenant`.

Returns

The updated tenant

Return type

Tenant

Identity API v3

For details on how to use identity, see *Using OpenStack Identity*

The Identity v3 Class

The identity high-level interface is available through the `identity` member of a *Connection* object. The `identity` member will only be added if the service is detected.

Credential Operations

class openstack.identity.v3._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_credential(***attrs*)

Create a new credential from attributes

Parameters

attrs (*dict*) Keyword arguments which will be used to create a *Credential*, comprised of the properties on the Credential class.

Returns

The results of credential creation

Return type

Credential

delete_credential(*credential*, *ignore_missing=True*)

Delete a credential

Parameters

- **credential** The value can be either the ID of a credential or a *Credential* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the credential does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent credential.

Returns

`None`

find_credential(*name_or_id*, *ignore_missing=True*)

Find a single credential

Parameters

- **name_or_id** The name or ID of a credential.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns

One *Credential* or None

get_credential(*credential*)

Get a single credential

Parameters

credential The value can be the ID of a credential or a *Credential* instance.

Returns

One *Credential*

Raises

NotFoundException when no resource can be found.

credentials(***query*)

Retrieve a generator of credentials

Parameters

query (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns

A generator of credentials instances.

Return type

Credential

update_credential(*credential*, ***attrs*)

Update a credential

Parameters

- **credential** Either the ID of a credential or a *Credential* instance.

- **attrs** The attributes to update on the credential represented by `credential`.

Returns
The updated credential

Return type
Credential

Domain Operations

class openstack.identity.v3._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_domain(***attrs*)

Create a new domain from attributes

Parameters
attrs (*dict*) Keyword arguments which will be used to create a *Domain*, comprised of the properties on the Domain class.

Returns
The results of domain creation

Return type
Domain

delete_domain(*domain*, *ignore_missing=True*)

Delete a domain

Parameters

- **domain** The value can be either the ID of a domain or a *Domain* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the domain does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent domain.

Returns
`None`

find_domain(*name_or_id*, *ignore_missing=True*)

Find a single domain

Parameters

- **name_or_id** The name or ID of a domain.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns
One *Domain* or None

get_domain(*domain*)

Get a single domain

Parameters
domain The value can be the ID of a domain or a *Domain* instance.

Returns
One *Domain*

Raises
NotFoundException when no resource can be found.

domains(***query*)

Retrieve a generator of domains

Parameters
query (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of domain instances.

Return type
Domain

update_domain(*domain*, ***attrs*)

Update a domain

Parameters

- **domain** Either the ID of a domain or a *Domain* instance.
- **attrs** The attributes to update on the domain represented by `domain`.

Returns
The updated domain

Return type
Domain

Domain Config Operations

class openstack.identity.v3._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_domain_config(*domain*, ***attrs*)

Create a new config for a domain from attributes.

Parameters

- **domain** The value can be the ID of a domain or a *Domain* instance.
- **attrs** (*dict*) Keyword arguments which will be used to create a *DomainConfig* comprised of the properties on the DomainConfig class.

Returns
The results of domain config creation

Return type
DomainConfig

`delete_domain_config`(*domain*, *ignore_missing=True*)

Delete a config for a domain

Parameters

- **domain** The value can be the ID of a domain or a a *Domain* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the identity provider does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent config for a domain.

Returns
`None`

`get_domain_config`(*domain*)

Get a single config for a domain

Parameters
domain_id The value can be the ID of a domain or a *Domain* instance.

Returns
One *DomainConfig*

Raises
NotFoundException when no resource can be found.

`update_domain_config`(*domain*, ***attrs*)

Update a config for a domain

Parameters

- **domain_id** The value can be the ID of a domain or a *Domain* instance.
- **attrs** The attributes to update on the config for a domain represented by `domain_id`.

Returns
The updated config for a domain

Return type
DomainConfig

Endpoint Operations

`class openstack.identity.v3._proxy.Proxy`(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_endpoint(***attrs*)

Create a new endpoint from attributes

Parameters

attrs (*dict*) Keyword arguments which will be used to create a *Endpoint*, comprised of the properties on the Endpoint class.

Returns

The results of endpoint creation

Return type

Endpoint

delete_endpoint(*endpoint*, *ignore_missing=True*)

Delete an endpoint

Parameters

- **endpoint** The value can be either the ID of an endpoint or a *Endpoint* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the endpoint does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent endpoint.

Returns

`None`

find_endpoint(*name_or_id*, *ignore_missing=True*)

Find a single endpoint

Parameters

- **name_or_id** The name or ID of a endpoint.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns

One *Endpoint* or None

get_endpoint(*endpoint*)

Get a single endpoint

Parameters

endpoint The value can be the ID of an endpoint or a *Endpoint* instance.

Returns

One *Endpoint*

Raises

NotFoundException when no resource can be found.

endpoints(***query*)

Retrieve a generator of endpoints

Parameters

query (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of endpoint instances.

Return type
Endpoint

update_endpoint(*endpoint*, ***attrs*)

Update a endpoint

Parameters

- **endpoint** Either the ID of an endpoint or a *Endpoint* instance.
- **attrs** The attributes to update on the endpoint represented by `endpoint`.

Returns
The updated endpoint

Return type
Endpoint

project_endpoints(*project*, ***query*)

Retrieve a generator of endpoints which are associated with the project.

Parameters

- **project** Either the project ID or an instance of *Project*
- **query** (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of endpoint instances.

Return type
`ProjectEndpoint`

associate_endpoint_with_project(*project*, *endpoint*)

Creates a direct association between project and endpoint

Parameters

- **project** Either the ID of a project or a *Project* instance.
- **endpoint** Either the ID of an endpoint or a *Endpoint* instance.

Returns
None

disassociate_endpoint_from_project(*project*, *endpoint*)

Removes a direct association between project and endpoint

Parameters

- **project** Either the ID of a project or a *Project* instance.
- **endpoint** Either the ID of an endpoint or a *Endpoint* instance.

Returns
None

Group Operations

class openstack.identity.v3._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_group(***attrs*)

Create a new group from attributes

Parameters

attrs (*dict*) Keyword arguments which will be used to create a *Group*, comprised of the properties on the Group class.

Returns

The results of group creation

Return type

Group

delete_group(*group*, *ignore_missing=True*)

Delete a group

Parameters

- **group** The value can be either the ID of a group or a *Group* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the group does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent group.

Returns

`None`

find_group(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single group

Parameters

- **name_or_id** The name or ID of a group.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns

One *Group* or None

get_group(*group*)

Get a single group

Parameters

group The value can be the ID of a group or a *Group* instance.

Returns

One *Group*

Raises

NotFoundException when no resource can be found.

groups(***query*)

Retrieve a generator of groups

Parameters

query (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns

A generator of group instances.

Return type

Group

update_group(*group*, ***attrs*)

Update a group

Parameters

- **group** Either the ID of a group or a *Group* instance.
- **attrs** The attributes to update on the group represented by `group`.

Returns

The updated group

Return type

Group

add_user_to_group(*user*, *group*)

Add user to group

Parameters

- **user** Either the ID of a user or a *User* instance.
- **group** Either the ID of a group or a *Group* instance.

Returns

`None`

remove_user_from_group(*user*, *group*)

Remove user to group

Parameters

- **user** Either the ID of a user or a *User* instance.
- **group** Either the ID of a group or a *Group* instance.

Returns

`None`

check_user_in_group(*user*, *group*)

Check whether user belongsto group

Parameters

- **user** Either the ID of a user or a *User* instance.
- **group** Either the ID of a group or a *Group* instance.

Returns
A boolean representing current relation

`group_users`(*group*, ***attrs*)
List users in a group

Parameters

- **group** Either the ID of a group or a *Group* instance.
- **attrs** Only password_expires_at can be filter for result.

Returns
List of *User*

Policy Operations

`class openstack.identity.v3._proxy.Proxy`(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

`create_policy`(***attrs*)
Create a new policy from attributes

Parameters
attrs (*dict*) Keyword arguments which will be used to create a *Policy*, comprised of the properties on the Policy class.

Returns
The results of policy creation

Return type
Policy

`delete_policy`(*policy*, *ignore_missing=True*)
Delete a policy

Parameters

- **policy** The value can be either the ID of a policy or a *Policy* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the policy does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent policy.

Returns
None

`find_policy`(*name_or_id*, *ignore_missing=True*)
Find a single policy

Parameters

- **name_or_id** The name or ID of a policy.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns
One *Policy* or None

`get_policy`(*policy*)

Get a single policy

Parameters
policy The value can be the ID of a policy or a *Policy* instance.

Returns
One *Policy*

Raises
NotFoundException when no resource can be found.

`policies`(***query*)

Retrieve a generator of policies

Parameters
query (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of policy instances.

Return type
Policy

`update_policy`(*policy*, ***attrs*)

Update a policy

Parameters

- **policy** Either the ID of a policy or a *Policy* instance.
- **attrs** The attributes to update on the policy represented by `policy`.

Returns
The updated policy

Return type
Policy

Project Operations

`class openstack.identity.v3._proxy.Proxy`(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

`create_project`(***attrs*)

Create a new project from attributes

Parameters
attrs (*dict*) Keyword arguments which will be used to create a *Project*, comprised of the properties on the Project class.

Returns
The results of project creation

Return type
Project

delete_project(*project*, *ignore_missing=True*)

Delete a project

Parameters

- **project** The value can be either the ID of a project or a *Project* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the project does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent project.

Returns
`None`

find_project(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single project

Parameters

- **name_or_id** The name or ID of a project.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns
One *Project* or None

get_project(*project*)

Get a single project

Parameters
project The value can be the ID of a project or a *Project* instance.

Returns
One *Project*

Raises
NotFoundException when no resource can be found.

projects(***query*)

Retrieve a generator of projects

Parameters
query (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of project instances.

Return type
Project

user_projects(*user*, ***query*)

Retrieve a generator of projects to which the user has authorization
to access.

Parameters

- **user** Either the user id or an instance of *User*
- **query** (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of project instances.

Return type
UserProject

endpoint_projects(*endpoint*, ***query*)

Retrieve a generator of projects which are associated with the endpoint.

Parameters

- **endpoint** Either the endpoint ID or an instance of *Endpoint*
- **query** (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of project instances.

Return type
EndpointProject

update_project(*project*, ***attrs*)

Update a project

Parameters

- **project** Either the ID of a project or a *Project* instance.
- **attrs** The attributes to update on the project represented by project.

Returns
The updated project

Return type
Project

Service Operations

class openstack.identity.v3._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_service(***attrs*)

Create a new service from attributes

Parameters

attrs (*dict*) Keyword arguments which will be used to create a *Service*, comprised of the properties on the Service class.

Returns

The results of service creation

Return type

Service

delete_service(*service*, *ignore_missing=True*)

Delete a service

Parameters

- **service** The value can be either the ID of a service or a *Service* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the service does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent service.

Returns

None

find_service(*name_or_id*, *ignore_missing=True*)

Find a single service

Parameters

- **name_or_id** The name or ID of a service.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns

One *Service* or None

get_service(*service*)

Get a single service

Parameters

service The value can be the ID of a service or a *Service* instance.

Returns

One *Service*

Raises

NotFoundException when no resource can be found.

services(***query*)

Retrieve a generator of services

Parameters

query (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of service instances.

Return type
Service

`update_service`(*service*, ***attrs*)

Update a service

Parameters

- **service** Either the ID of a service or a *Service* instance.
- **attrs** The attributes to update on the service represented by `service`.

Returns
The updated service

Return type
Service

User Operations

`class openstack.identity.v3._proxy.Proxy`(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

`create_user`(***attrs*)

Create a new user from attributes

Parameters
attrs (*dict*) Keyword arguments which will be used to create a *User*, comprised of the properties on the User class.

Returns
The results of user creation

Return type
User

`delete_user`(*user*, *ignore_missing=True*)

Delete a user

Parameters

- **user** The value can be either the ID of a user or a *User* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the user does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent user.

Returns
`None`

`find_user`(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single user

Parameters

- **name_or_id** The name or ID of a user.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns
One *User* or None

get_user(*user*)

Get a single user

Parameters
user The value can be the ID of a user or a *User* instance.

Returns
One *User*

Raises
NotFoundException when no resource can be found.

users(***query*)

Retrieve a generator of users

Parameters
query (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of user instances.

Return type
User

update_user(*user*, ***attrs*)

Update a user

Parameters

- **user** Either the ID of a user or a *User* instance.
- **attrs** The attributes to update on the user represented by `attrs`.

Returns
The updated user

Return type
User

Trust Operations

class openstack.identity.v3._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_trust(***attrs*)

Create a new trust from attributes

Parameters

attrs (*dict*) Keyword arguments which will be used to create a *Trust*, comprised of the properties on the Trust class.

Returns

The results of trust creation

Return type

Trust

delete_trust(*trust*, *ignore_missing=True*)

Delete a trust

Parameters

- **trust** The value can be either the ID of a trust or a *Trust* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the credential does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent credential.

Returns

`None`

find_trust(*name_or_id*, *ignore_missing=True*)

Find a single trust

Parameters

- **name_or_id** The name or ID of a trust.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns

One *Trust* or None

get_trust(*trust*)

Get a single trust

Parameters

trust The value can be the ID of a trust or a *Trust* instance.

Returns

One *Trust*

Raises

NotFoundException when no resource can be found.

trusts(***query*)

Retrieve a generator of trusts

Parameters

query (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of trust instances.

Return type
Trust

Region Operations

class openstack.identity.v3._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_region(***attrs*)

Create a new region from attributes

Parameters
attrs (*dict*) Keyword arguments which will be used to create a *Region*, comprised of the properties on the Region class.

Returns
The results of region creation.

Return type
Region

delete_region(*region*, *ignore_missing=True*)

Delete a region

Parameters

- **region** The value can be either the ID of a region or a *Region* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the region does not exist. When set to `True`, no exception will be thrown when attempting to delete a nonexistent region.

Returns
None

find_region(*name_or_id*, *ignore_missing=True*)

Find a single region

Parameters

- **name_or_id** The name or ID of a region.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the region does not exist. When set to `True`, None will be returned when attempting to find a nonexistent region.

Returns
One *Region* or None

get_region(*region*)

Get a single region

Parameters
region The value can be the ID of a region or a *Region* instance.

Returns
One *Region*

Raises
NotFoundException when no matching region can be found.

regions(***query*)

Retrieve a generator of regions

Parameters
query (*kwargs*) Optional query parameters to be sent to limit the regions being returned.

Returns
A generator of region instances.

Return type
Region

update_region(*region*, ***attrs*)

Update a region

Parameters

- **region** Either the ID of a region or a *Region* instance.
- **attrs** The attributes to update on the region represented by `region`.

Returns
The updated region.

Return type
Region

Role Operations

class openstack.identity.v3._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_role(***attrs*)

Create a new role from attributes

Parameters
attrs (*dict*) Keyword arguments which will be used to create a *Role*, comprised of the properties on the Role class.

Returns
The results of role creation.

Return type
Role

delete_role(*role*, *ignore_missing=True*)

Delete a role

Parameters

- **role** The value can be either the ID of a role or a *Role* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the role does not exist. When set to `True`, no exception will be thrown when attempting to delete a nonexistent role.

Returns

`None`

find_role(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single role

Parameters

- **name_or_id** The name or ID of a role.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the role does not exist. When set to `True`, None will be returned when attempting to find a nonexistent role.

Returns

One *Role* or None

get_role(*role*)

Get a single role

Parameters

role The value can be the ID of a role or a *Role* instance.

Returns

One *Role*

Raises

NotFoundException when no matching role can be found.

roles(***query*)

Retrieve a generator of roles

Parameters

query (*kwargs*) Optional query parameters to be sent to limit the resources being returned. The options are: domain_id, name.

Returns

A generator of role instances.

Return type

Role

update_role(*role*, ***attrs*)

Update a role

Parameters

- **role** Either the ID of a role or a *Role* instance.

- **kwargs** (*dict*) The attributes to update on the role represented by `value`. Only name can be updated

Returns
The updated role.

Return type
Role

Role Assignment Operations

class openstack.identity.v3._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

role_assignments_filter(*domain=None*, *project=None*, *system=None*, *group=None*, *user=None*)

Retrieve a generator of roles assigned to user/group

Parameters

- **domain** Either the ID of a domain or a *Domain* instance.
- **project** Either the ID of a project or a *Project* instance.
- **system** Either the system name or a *System* instance.
- **group** Either the ID of a group or a *Group* instance.
- **user** Either the ID of a user or a *User* instance.

Returns
A generator of role instances.

Return type
Role

role_assignments(***query*)

Retrieve a generator of role assignments

Parameters
query (*kwargs*) Optional query parameters to be sent to limit the resources being returned. The options are: group_id, role_id, scope_domain_id, scope_project_id, inherited_to, user_id, include_names, include_subtree.

Returns
RoleAssignment

assign_domain_role_to_user(*domain*, *user*, *role*, ***, *inherited=False*)

Assign role to user on a domain

Parameters

- **domain** Either the ID of a domain or a *Domain* instance.
- **user** Either the ID of a user or a *User* instance.
- **role** Either the ID of a role or a *Role* instance.

- **inherited** (*bool*) Whether the role assignment is inherited.

Returns
None

unassign_domain_role_from_user(*domain*, *user*, *role*, *, *inherited=False*)

Unassign role from user on a domain

Parameters

- **domain** Either the ID of a domain or a *Domain* instance.
- **user** Either the ID of a user or a *User* instance.
- **role** Either the ID of a role or a *Role* instance.
- **inherited** (*bool*) Whether the role assignment is inherited.

Returns
None

validate_user_has_domain_role(*domain*, *user*, *role*, *, *inherited=False*)

Validates that a user has a role on a domain

Parameters

- **domain** Either the ID of a domain or a *Domain* instance.
- **user** Either the ID of a user or a *User* instance.
- **role** Either the ID of a role or a *Role* instance.

Returns
True if user has role in domain

assign_domain_role_to_group(*domain*, *group*, *role*, *, *inherited=False*)

Assign role to group on a domain

Parameters

- **domain** Either the ID of a domain or a *Domain* instance.
- **group** Either the ID of a group or a *Group* instance.
- **role** Either the ID of a role or a *Role* instance.
- **inherited** (*bool*) Whether the role assignment is inherited.

Returns
None

unassign_domain_role_from_group(*domain*, *group*, *role*, *, *inherited=False*)

Unassign role from group on a domain

Parameters

- **domain** Either the ID of a domain or a *Domain* instance.
- **group** Either the ID of a group or a *Group* instance.
- **role** Either the ID of a role or a *Role* instance.
- **inherited** (*bool*) Whether the role assignment is inherited.

Returns
None

validate_group_has_domain_role(*domain*, *group*, *role*, *, *inherited=False*)

Validates that a group has a role on a domain

Parameters

- **domain** Either the ID of a domain or a *Domain* instance.
- **group** Either the ID of a group or a *Group* instance.
- **role** Either the ID of a role or a *Role* instance.

Returns
True if group has role on domain

assign_project_role_to_user(*project*, *user*, *role*, *, *inherited=False*)

Assign role to user on a project

Parameters

- **project** Either the ID of a project or a *Project* instance.
- **user** Either the ID of a user or a *User* instance.
- **role** Either the ID of a role or a *Role* instance.
- **inherited** (*bool*) Whether the role assignment is inherited.

Returns
None

unassign_project_role_from_user(*project*, *user*, *role*, *, *inherited=False*)

Unassign role from user on a project

Parameters

- **project** Either the ID of a project or a *Project* instance.
- **user** Either the ID of a user or a *User* instance.
- **role** Either the ID of a role or a *Role* instance.
- **inherited** (*bool*) Whether the role assignment is inherited.

Returns
None

validate_user_has_project_role(*project*, *user*, *role*, *, *inherited=False*)

Validates that a user has a role on a project

Parameters

- **project** Either the ID of a project or a *Project* instance.
- **user** Either the ID of a user or a *User* instance.
- **role** Either the ID of a role or a *Role* instance.

Returns
True if user has role in project

assign_project_role_to_group(*project*, *group*, *role*, *, *inherited=False*)

Assign role to group on a project

Parameters

- **project** Either the ID of a project or a *Project* instance.
- **group** Either the ID of a group or a *Group* instance.
- **role** Either the ID of a role or a *Role* instance.
- **inherited** (*bool*) Whether the role assignment is inherited.

Returns

None

unassign_project_role_from_group(*project*, *group*, *role*, *, *inherited=False*)

Unassign role from group on a project

Parameters

- **project** Either the ID of a project or a *Project* instance.
- **group** Either the ID of a group or a *Group* instance.
- **role** Either the ID of a role or a *Role* instance.
- **inherited** (*bool*) Whether the role assignment is inherited.

Returns

None

validate_group_has_project_role(*project*, *group*, *role*, *, *inherited=False*)

Validates that a group has a role on a project

Parameters

- **project** Either the ID of a project or a *Project* instance.
- **group** Either the ID of a group or a *Group* instance.
- **role** Either the ID of a role or a *Role* instance.

Returns

True if group has role in project

assign_system_role_to_user(*user*, *role*, *system*)

Assign a role to user on a system

Parameters

- **user** Either the ID of a user or a *User* instance.
- **role** Either the ID of a role or a *Role* instance.
- **system** The system name

Returns

None

unassign_system_role_from_user(*user*, *role*, *system*)

Unassign a role from user on a system

Parameters

- **user** Either the ID of a user or a *User* instance.
- **role** Either the ID of a role or a *Role* instance.
- **system** The system name

Returns
None

validate_user_has_system_role(*user*, *role*, *system*)

Validates that a user has a role on a system

Parameters

- **user** Either the ID of a user or a *User* instance.
- **role** Either the ID of a role or a *Role* instance.
- **system** The system name

Returns
True if user has role in system

assign_system_role_to_group(*group*, *role*, *system*)

Assign a role to group on a system

Parameters

- **group** Either the ID of a group or a *Group* instance.
- **role** Either the ID of a role or a *Role* instance.
- **system** The system name

Returns
None

unassign_system_role_from_group(*group*, *role*, *system*)

Unassign a role from group on a system

Parameters

- **group** Either the ID of a group or a *Group* instance.
- **role** Either the ID of a role or a *Role* instance.
- **system** The system name

Returns
None

validate_group_has_system_role(*group*, *role*, *system*)

Validates that a group has a role on a system

Parameters

- **group** Either the ID of a group or a *Group* instance.
- **role** Either the ID of a role or a *Role* instance.
- **system** The system name

Returns
True if group has role on system

Registered Limit Operations

class openstack.identity.v3._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

registered_limits(***query*)

Retrieve a generator of registered_limits

Parameters

query (*kwargs*) Optional query parameters to be sent to limit the registered_limits being returned.

Returns

A generator of registered_limits instances.

Return type

RegisteredLimit

get_registered_limit(*registered_limit*)

Get a single registered_limit

Parameters

registered_limit The value can be the ID of a registered_limit or a *RegisteredLimit* instance.

Returns

One *RegisteredLimit*

Raises

NotFoundException when no resource can be found.

create_registered_limit(***attrs*)

Create a new registered_limit from attributes

Parameters

attrs (*dict*) Keyword arguments which will be used to create a *RegisteredLimit*, comprised of the properties on the RegisteredLimit class.

Returns

The results of registered_limit creation.

Return type

RegisteredLimit

update_registered_limit(*registered_limit*, ***attrs*)

Update a registered_limit

Parameters

- **registered_limit** Either the ID of a registered_limit. or a *RegisteredLimit* instance.
- **kwargs** (*dict*) The attributes to update on the registered_limit represented by `value`.

Returns
The updated registered_limit.

Return type
RegisteredLimit

`delete_registered_limit`(*registered_limit*, *ignore_missing=True*)

Delete a registered_limit

Parameters

- **registered_limit** The value can be either the ID of a registered_limit or a *RegisteredLimit* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the registered_limit does not exist. When set to `True`, no exception will be thrown when attempting to delete a nonexistent registered_limit.

Returns
`None`

Limit Operations

`class openstack.identity.v3._proxy.Proxy`(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

`limits`(***query*)

Retrieve a generator of limits

Parameters
query (*kwargs*) Optional query parameters to be sent to limit the limits being returned.

Returns
A generator of limits instances.

Return type
Limit

`get_limit`(*limit*)

Get a single limit

Parameters
limit The value can be the ID of a limit or a *Limit* instance.

Returns
One *Limit*

Raises
NotFoundException when no resource can be found.

`create_limit`(***attrs*)

Create a new limit from attributes

Parameters
attrs (*dict*) Keyword arguments which will be used to create a *Limit*, comprised of the properties on the Limit class.

Returns
The results of limit creation.

Return type
Limit

update_limit(*limit*, ***attrs*)

Update a limit

Parameters

- **limit** Either the ID of a limit. or a *Limit* instance.
- **kwargs** (*dict*) The attributes to update on the limit represented by `value`.

Returns
The updated limit.

Return type
Limit

delete_limit(*limit*, *ignore_missing=True*)

Delete a limit

Parameters

- **limit** The value can be either the ID of a limit or a *Limit* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the limit does not exist. When set to `True`, no exception will be thrown when attempting to delete a nonexistent limit.

Returns
`None`

Application Credential Operations

class openstack.identity.v3._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

application_credentials(*user*, ***query*)

Retrieve a generator of application credentials

Parameters

- **user** Either the ID of a user or a *User* instance.
- **query** (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of application credentials instances.

Return type
ApplicationCredential

get_application_credential(*user*, *application_credential*)

Get a single application credential

Parameters

- **user** Either the ID of a user or a *User* instance.
- **application_credential** The value can be the ID of a application credential or a *ApplicationCredential* instance.

Returns
One *ApplicationCredential*

Raises
NotFoundException when no resource can be found.

create_application_credential(*user*, *name*, ***attrs*)

Create a new application credential from attributes

Parameters

- **user** Either the ID of a user or a *User* instance.
- **name** The name of the application credential which is unique to the user.
- **attrs** (*dict*) Keyword arguments which will be used to create a *ApplicationCredential*, comprised of the properties on the ApplicationCredential class.

Returns
The results of application credential creation.

Return type
ApplicationCredential

find_application_credential(*user*, *name_or_id*, *ignore_missing=True*, ***query*)

Find a single application credential

Parameters

- **user** Either the ID of a user or a *User* instance.
- **name_or_id** The name or ID of an application credential.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns
One *ApplicationCredential* or None

delete_application_credential(*user*, *application_credential*, *ignore_missing=True*)

Delete an application credential

Parameters

- **user** Either the ID of a user or a *User* instance.

- **credential** (*application*) The value can be either the ID of an application credential or a *ApplicationCredential* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the application credential does not exist. When set to `True`, no exception will be thrown when attempting to delete a nonexistent application credential.

Returns
`None`

Federation Protocol Operations

`class openstack.identity.v3._proxy.Proxy(session, statsd_client=None, statsd_prefix=None, prometheus_counter=None, prometheus_histogram=None, influxdb_config=None, influxdb_client=None, *args, **kwargs)`

`create_federation_protocol(idp_id, **attrs)`

Create a new federation protocol from attributes

Parameters

- **idp_id** The ID of the identity provider or a *IdentityProvider* representing the identity provider the protocol is to be attached to.
- **attrs** (*dict*) Keyword arguments which will be used to create a *FederationProtocol*, comprised of the properties on the FederationProtocol class.

Returns
The results of federation protocol creation

Return type
FederationProtocol

`delete_federation_protocol(idp_id, protocol, ignore_missing=True)`

Delete a federation protocol

Parameters

- **idp_id** The ID of the identity provider or a *IdentityProvider* representing the identity provider the protocol is attached to. Can be None if protocol is a *FederationProtocol* instance.
- **protocol** The ID of a federation protocol or a *FederationProtocol* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the federation protocol does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent federation protocol.

Returns
`None`

find_federation_protocol(*idp_id*, *protocol*, *ignore_missing=True*)

Find a single federation protocol

Parameters

- **idp_id** The ID of the identity provider or a *IdentityProvider* representing the identity provider the protocol is attached to.
- **protocol** The name or ID of a federation protocol.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns

One federation protocol or None

Return type

FederationProtocol

get_federation_protocol(*idp_id*, *protocol*)

Get a single federation protocol

Parameters

- **idp_id** The ID of the identity provider or a *IdentityProvider* representing the identity provider the protocol is attached to. Can be None if protocol is a *FederationProtocol*
- **protocol** The value can be the ID of a federation protocol or a *FederationProtocol* instance.

Returns

One federation protocol

Return type

FederationProtocol

Raises

NotFoundException when no resource can be found.

federation_protocols(*idp_id*, ***query*)

Retrieve a generator of federation protocols

Parameters

- **idp_id** The ID of the identity provider or a *IdentityProvider* representing the identity provider the protocol is attached to.
- **query** (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns

A generator of federation protocol instances.

Return type

FederationProtocol

update_federation_protocol(*idp_id*, *protocol*, ***attrs*)

Update a federation protocol

Parameters

- **idp_id** The ID of the identity provider or a *IdentityProvider* representing the identity provider the protocol is attached to. Can be None if protocol is a *FederationProtocol*
- **protocol** Either the ID of a federation protocol or a *FederationProtocol* instance.
- **attrs** The attributes to update on the federation protocol represented by `protocol`.

Returns
The updated federation protocol

Return type
FederationProtocol

Mapping Operations

class openstack.identity.v3._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_mapping(***attrs*)

Create a new mapping from attributes

Parameters
attrs (*dict*) Keyword arguments which will be used to create a *Mapping*, comprised of the properties on the Mapping class.

Returns
The results of mapping creation

Return type
Mapping

delete_mapping(*mapping*, *ignore_missing=True*)

Delete a mapping

Parameters

- **mapping** The ID of a mapping or a *Mapping* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the mapping does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent mapping.

Returns
None

find_mapping(*name_or_id*, *ignore_missing=True*)

Find a single mapping

Parameters

- **name_or_id** The name or ID of a mapping.

- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns
One *Mapping* or None

get_mapping(*mapping*)

Get a single mapping

Parameters
mapping The value can be the ID of a mapping or a *Mapping* instance.

Returns
One *Mapping*

Raises
NotFoundException when no resource can be found.

mappings(***query*)

Retrieve a generator of mappings

Parameters
query (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of mapping instances.

Return type
Mapping

update_mapping(*mapping*, ***attrs*)

Update a mapping

Parameters

- **mapping** Either the ID of a mapping or a *Mapping* instance.
- **attrs** The attributes to update on the mapping represented by `mapping`.

Returns
The updated mapping

Return type
Mapping

Identity Provider Operations

class openstack.identity.v3._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_identity_provider(***attrs*)

Create a new identity provider from attributes

Parameters
attrs (*dict*) Keyword arguments which will be used to create a *IdentityProvider* comprised of the properties on the IdentityProvider class.

Returns
The results of identity provider creation

Return type
IdentityProvider

delete_identity_provider(*identity_provider*, *ignore_missing=True*)

Delete an identity provider

Parameters

- **mapping** The ID of an identity provoder or a *IdentityProvider* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the identity provider does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent identity provider.

Returns
`None`

find_identity_provider(*name_or_id*, *ignore_missing=True*)

Find a single identity provider

Parameters

- **name_or_id** The name or ID of an identity provider
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns
The details of an identity provider or None.

Return type
IdentityProvider

get_identity_provider(*identity_provider*)

Get a single mapping

Parameters
mapping The value can be the ID of an identity provider or a *IdentityProvider* instance.

Returns
The details of an identity provider.

Return type
IdentityProvider

Raises
NotFoundException when no resource can be found.

identity_providers(***query*)

Retrieve a generator of identity providers

Parameters
query (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of identity provider instances.

Return type
IdentityProvider

update_identity_provider(*identity_provider*, ***attrs*)

Update a mapping

Parameters

- **mapping** Either the ID of an identity provider or a *IdentityProvider* instance.
- **attrs** The attributes to update on the identity_provider represented by `identity_provider`.

Returns
The updated identity provider.

Return type
IdentityProvider

Access Rule Operations

class openstack.identity.v3._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

access_rules(*user*, ***query*)

Retrieve a generator of access rules

Parameters

- **user** Either the ID of a user or a *User* instance.
- **query** (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of access rules instances.

Return type
`AccessRule`

access_rules(*user*, ***query*)

Retrieve a generator of access rules

Parameters

- **user** Either the ID of a user or a *User* instance.
- **query** (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of access rules instances.

Return type
AccessRule

delete_access_rule(*user*, *access_rule*, *ignore_missing=True*)

Delete an access rule

Parameters

- **user** Either the ID of a user or a *User* instance.
- **rule** (*access*) The value can be either the ID of an access rule or a AccessRule instance.
- **ignore_missing** (*bool*) When set to False *NotFoundException* will be raised when the access rule does not exist. When set to True, no exception will be thrown when attempting to delete a nonexistent access rule.

Returns
None

Service Provider Operations

class openstack.identity.v3._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_service_provider(***attrs*)

Create a new service provider from attributes

Parameters
attrs (*dict*) Keyword arguments which will be used to create a ServiceProvider, comprised of the properties on the ServiceProvider class.

Returns
The results of service provider creation

Return type
ServiceProvider

delete_service_provider(*service_provider*, *ignore_missing=True*)

Delete a service provider

Parameters

- **service_provider** The ID of a service provider or a ServiceProvider instance.
- **ignore_missing** (*bool*) When set to False *NotFoundException* will be raised when the service provider does not exist. When set to True, no exception will be set when attempting to delete a nonexistent service provider.

Returns
None

find_service_provider(*name_or_id*, *ignore_missing=True*)

Find a single service provider

Parameters

- **name_or_id** The name or ID of a service provider
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns
The details of an service provider or None.

Return type
`ServiceProvider`

get_service_provider(*service_provider*)

Get a single service provider

Parameters
service_provider The value can be the ID of a service provider or a `ServiceProvider` instance.

Returns
The details of an service provider.

Return type
`ServiceProvider`

Raises
NotFoundException when no resource can be found.

service_providers(***query*)

Retrieve a generator of service providers

Parameters
query (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of service provider instances.

Return type
`ServiceProvider`

update_service_provider(*service_provider*, ***attrs*)

Update a service provider

Parameters

- **service_provider** Either the ID of an service provider or a `ServiceProvider` instance.
- **attrs** The attributes to update on the service provider represented by `service_provider`.

Returns
The updated service provider.

Return type
ServiceProvider

Image API v1

For details on how to use image, see *Using OpenStack Image*

The Image v1 Class

The image high-level interface is available through the `image` member of a *Connection* object. The `image` member will only be added if the service is detected.

class openstack.image.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

upload_image(***attrs*)

Upload a new image from attributes

Parameters

attrs (*dict*) Keyword arguments which will be used to create a *Image*, comprised of the properties on the Image class.

Returns

The results of image creation

Return type

Image

delete_image(*image*, *ignore_missing=True*)

Delete an image

Parameters

- **image** The value can be either the ID of an image or a *Image* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the image does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent image.

Returns

None

find_image(*name_or_id*, *ignore_missing=True*)

Find a single image

Parameters

- **name_or_id** The name or ID of a image.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns

One *Image* or None

get_image(*image*)

Get a single image

Parameters

image The value can be the ID of an image or a *Image* instance.

Returns

One *Image*

Raises

NotFoundException when no resource can be found.

images(***query*)

Return a generator of images

Parameters

query (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns

A generator of image objects

Return type

Image

update_image(*image*, ***attrs*)

Update a image

Parameters

- **image** Either the ID of a image or a *Image* instance.
- **attrs** The attributes to update on the image represented by `image`.

Returns

The updated image

Return type

Image

Image API v2

For details on how to use image, see *Using OpenStack Image*

The Image v2 Class

The image high-level interface is available through the `image` member of a *Connection* object. The `image` member will only be added if the service is detected.

Image Operations

class openstack.image.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_image(*name*, *, *filename=None*, *data=None*, *container=None*, *md5=None*, *sha256=None*, *disk_format=None*, *container_format=None*, *tags=None*, *disable_vendor_agent=True*, *allow_duplicates=False*, *meta=None*, *wait=False*, *timeout=3600*, *validate_checksum=False*, *use_import=False*, *stores=None*, *all_stores=None*, *all_stores_must_succeed=None*, ***kwargs*)

Create an image and optionally upload data

Create a new image. If `filename` or `data` are provided, it will also upload data to this image.

Note that uploading image data is actually quite a complicated procedure. There are three ways to upload an image:

- Image upload
- Image import
- Image tasks

If the image tasks API is enabled, this must be used. However, this API is deprecated since the Image services Mitaka (12.0.0) release and is now admin-only. Assuming this API is not enabled, you may choose between image upload or image import. Image import is more powerful and allows you to upload data from multiple sources including other glance instances. It should be preferred on all services that support it.

Parameters

- **name** (*str*) Name of the image to create. If it is a pathname of an image, the name will be constructed from the extensionless basename of the path.
- **filename** (*str*) The path to the file to upload, if needed. (optional, defaults to None)
- **data** Image data (string or file-like object). It is mutually exclusive with filename
- **container** (*str*) Name of the container in swift where images should be uploaded for import if the cloud requires such a thing. (optional, defaults to images)
- **md5** (*str*) md5 sum of the image file. If not given, an md5 will be calculated.
- **sha256** (*str*) sha256 sum of the image file. If not given, an md5 will be calculated.
- **disk_format** (*str*) The disk format the image is in. (optional, defaults to the os-client-config config value for this cloud)
- **container_format** (*str*) The container format the image is in. (optional, defaults to the os-client-config config value for this cloud)
- **tags** (*list*) List of tags for this image. Each tag is a string of at most 255 chars.
- **disable_vendor_agent** (*bool*) Whether or not to append metadata flags to the image to inform the cloud in question to not expect a vendor agent to be runing. (optional, defaults to True)
- **allow_duplicates** If true, skips checks that enforce unique image name. (optional, defaults to False)

- **meta** A dict of key/value pairs to use for metadata that bypasses automatic type conversion.
- **wait** (*bool*) If true, waits for image to be created. Defaults to true - however, be aware that one of the upload methods is always synchronous.
- **timeout** Seconds to wait for image creation. None is forever.
- **validate_checksum** (*bool*) If true and cloud returns checksum, compares return value with the one calculated or passed into this call. If value does not match - raises exception. Default is false
- **use_import** (*bool*) Use the glance-direct method of the interoperable image import mechanism to import the image. This defaults to false because it is harder on the target cloud so should only be used when needed, such as when the user needs the cloud to transform image format. If the cloud has disabled direct uploads, this will default to true. If you wish to use other import methods, use the `import_image` method instead.
- **stores** List of stores to be used when enabled_backends is activated in glance. List values can be the id of a store or a *Store* instance. Implies `use_import` equals `True`.
- **all_stores** Upload to all available stores. Mutually exclusive with `store` and `stores`. Implies `use_import` equals `True`.
- **all_stores_must_succeed** When set to True, if an error occurs during the upload in at least one store, the worfklow fails, the data is deleted from stores where copying is done (not staging), and the state of the image is unchanged. When set to False, the workflow will fail (data deleted from stores,) only if the import fails on all stores specified by the user. In case of a partial success, the locations added to the image will be the stores where the data has been correctly uploaded. Default is True. Implies `use_import` equals `True`.

Additional kwargs will be passed to the image creation as additional metadata for the image and will have all values converted to string except for min_disk, min_ram, size and virtual_size which will be converted to int.

If you are sure you have all of your data types correct or have an advanced need to be explicit, use meta. If you are just a normal consumer, using kwargs is likely the right choice.

If a value is in meta and kwargs, meta wins.

Returns
The results of image creation

Return type
Image

Raises
SDKException if there are problems uploading

import_image(*image*, *method='glance-direct'*, ***, *uri=None*, *remote_region=None*, *remote_image_id=None*, *remote_service_interface=None*, *store=None*, *stores=None*, *all_stores=None*, *all_stores_must_succeed=None*)

Import data to an existing image

Interoperable image import process are introduced in the Image API v2.6. It mainly allow image importing from an external url and let Image Service download it by itself without sending binary data at image creation.

Parameters

- **image** The value can be the ID of a image or a *Image* instance.
- **method** Method to use for importing the image. Not all deployments support all methods. One of: `glance-direct` (default), `web-download`, `glance-download`, or `copy-image`. Use of `glance-direct` requires the image be first staged.
- **uri** Required only if using the `web-download` import method. This url is where the data is made available to the Image service.
- **remote_region** The remote glance region to download the image from when using glance-download.
- **remote_image_id** The ID of the image to import from the remote glance when using glance-download.
- **remote_service_interface** The remote glance service interface to use when using glance-download.
- **store** Used when enabled_backends is activated in glance. The value can be the id of a store or a. *Store* instance.
- **stores** List of stores to be used when enabled_backends is activated in glance. List values can be the id of a store or a *Store* instance.
- **all_stores** Upload to all available stores. Mutually exclusive with `store` and `stores`.
- **all_stores_must_succeed** When set to True, if an error occurs during the upload in at least one store, the worfklow fails, the data is deleted from stores where copying is done (not staging), and the state of the image is unchanged. When set to False, the workflow will fail (data deleted from stores,) only if the import fails on all stores specified by the user. In case of a partial success, the locations added to the image will be the stores where the data has been correctly uploaded. Default is True.

Returns

The raw response from the request.

stage_image(*image*, ***, *filename=None*, *data=None*)

Stage binary image data

Parameters

- **image** The value can be the ID of a image or a *Image* instance.
- **filename** Optional name of the file to read data from.
- **data** Optional data to be uploaded as an image.

Returns

The results of image creation

Return type

Image

upload_image(*container_format=None*, *disk_format=None*, *data=None*, ***attrs*)

Create and upload a new image from attributes

Parameters

- **container_format** Format of the container. A valid value is ami, ari, aki, bare, ovf, ova, or docker.
- **disk_format** The format of the disk. A valid value is ami, ari, aki, vhd, vmdk, raw, qcow2, vdi, or iso.
- **data** The data to be uploaded as an image.
- **attrs** (*dict*) Keyword arguments which will be used to create a *Image*, comprised of the properties on the Image class.

Returns

The results of image creation

Return type

Image

download_image(*image*, ***, *stream=False*, *output=None*, *chunk_size=1048576*)

Download an image

This will download an image to memory when `stream=False`, or allow streaming downloads using an iterator when `stream=True`. For examples of working with streamed responses, see *Downloading an Image with stream=True*.

Parameters

- **image** The value can be either the ID of an image or a *Image* instance.
- **stream** (*bool*) When `True`, return a `requests.Response` instance allowing you to iterate over the response data stream instead of storing its entire contents in memory. See `requests.Response.iter_content()` for more details.

  NOTE: If you do not consume the entirety of the response you must explicitly call `requests.Response.close()` or otherwise risk inefficiencies with the `requests` librarys handling of connections.

  When `False`, return the entire contents of the response.
- **output** Either a file object or a path to store data into.
- **chunk_size** (*int*) size in bytes to read from the wire and buffer at one time. Defaults to 1024 * 1024 = 1 MiB

Returns

When output is not given - the bytes comprising the given Image when stream is False, otherwise a `requests.Response` instance. When output is given - a *Image* instance.

delete_image(*image*, ***, *store=None*, *ignore_missing=True*)

Delete an image

Parameters

- **image** The value can be either the ID of an image or a *Image* instance.

- **store** The value can be either the ID of a store or a *Store* instance that the image is associated with. If specified, the image will only be deleted from the specified store.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the image does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent image.

Returns
`None`

find_image(*name_or_id*, *ignore_missing=True*)

Find a single image

Parameters

- **name_or_id** The name or ID of a image.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns
One *Image* or None

get_image(*image*)

Get a single image

Parameters
image The value can be the ID of a image or a *Image* instance.

Returns
One *Image*

Raises
NotFoundException when no resource can be found.

images(***query*)

Return a generator of images

Parameters
query (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of image objects

Return type
Image

update_image(*image*, ***attrs*)

Update a image

Parameters

- **image** Either the ID of a image or a *Image* instance.
- **attrs** The attributes to update on the image represented by `image`.

Returns
The updated image

Return type
Image

deactivate_image(*image*)

Deactivate an image

Parameters
image Either the ID of a image or a *Image* instance.

Returns
None

reactivate_image(*image*)

Reactivate an image

Parameters
image Either the ID of a image or a *Image* instance.

Returns
None

add_tag(*image*, *tag*)

Add a tag to an image

Parameters

- **image** The value can be the ID of a image or a *Image* instance that the member will be created for.
- **tag** (*str*) The tag to be added

Returns
None

remove_tag(*image*, *tag*)

Remove a tag to an image

Parameters

- **image** The value can be the ID of a image or a *Image* instance that the member will be created for.
- **tag** (*str*) The tag to be removed

Returns
None

Member Operations

class openstack.image.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

add_member(*image*, ***attrs*)

Create a new member from attributes

Parameters

- **image** The value can be the ID of a image or a *Image* instance that the member will be created for.
- **attrs** (*dict*) Keyword arguments which will be used to create a *Member*, comprised of the properties on the Member class.

See Image Sharing Reference for details.

Returns
The results of member creation

Return type
Member

remove_member(*member*, *image=None*, *ignore_missing=True*)

Delete a member

Parameters

- **member** The value can be either the ID of a member or a *Member* instance.
- **image** The value can be either the ID of an image or a *Image* instance that the member is part of. This is required if `member` is an ID.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the member does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent member.

Returns
`None`

find_member(*name_or_id*, *image*, *ignore_missing=True*)

Find a single member

Parameters

- **name_or_id** The name or ID of a member.
- **image** This is the image that the member belongs to, the value can be the ID of a image or a *Image* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns
One *Member* or None

get_member(*member*, *image*)

Get a single member on an image

Parameters

- **member** The value can be the ID of a member or a *Member* instance.
- **image** This is the image that the member belongs to. The value can be the ID of a image or a *Image* instance.

Returns
One *Member*

Raises
NotFoundException when no resource can be found.

members(*image*, ***query*)
Return a generator of members

Parameters

- **image** This is the image that the member belongs to, the value can be the ID of a image or a *Image* instance.
- **query** (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of member objects

Return type
Member

update_member(*member*, *image*, ***attrs*)
Update the member of an image

Parameters

- **member** Either the ID of a member or a *Member* instance.
- **image** This is the image that the member belongs to. The value can be the ID of a image or a *Image* instance.
- **attrs** The attributes to update on the member represented by `member`.

See Image Sharing Reference for details.

Returns
The updated member

Return type
Member

Task Operations

class openstack.image.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

tasks(***query*)
Return a generator of tasks

Parameters
query (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of task objects

Return type
Task

get_task(*task*)

Get task details

Parameters
task The value can be the ID of a task or a *Task* instance.

Returns
One *Task*

Raises
NotFoundException when no resource can be found.

create_task(***attrs*)

Create a new task from attributes

Parameters
attrs (*dict*) Keyword arguments which will be used to create a *Task*, comprised of the properties on the Task class.

Returns
The results of task creation

Return type
Task

wait_for_task(*task*, *status='success'*, *failures=None*, *interval=2*, *wait=120*)

Wait for a task to be in a particular status.

Parameters

- **task** The resource to wait on to reach the specified status. The resource must have a `status` attribute.
- **status** Desired status.
- **failures** (`list`) Statuses that would be interpreted as failures.
- **interval** Number of seconds to wait before to consecutive checks. Default to 2.
- **wait** Maximum number of seconds to wait before the change. Default to 120.

Returns
The resource is returned on success.

Raises
ResourceTimeout if transition to the desired status failed to occur in specified seconds.

Raises
ResourceFailure if the resource has transited to one of the failure statuses.

Raises
`AttributeError` if the resource does not have a `status` attribute.

Schema Operations

class openstack.image.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

get_images_schema()

Get images schema

Returns
One Schema

Raises
NotFoundException when no resource can be found.

get_image_schema()

Get single image schema

Returns
One Schema

Raises
NotFoundException when no resource can be found.

get_members_schema()

Get image members schema

Returns
One Schema

Raises
NotFoundException when no resource can be found.

get_member_schema()

Get image member schema

Returns
One Schema

Raises
NotFoundException when no resource can be found.

get_tasks_schema()

Get image tasks schema

Returns
One Schema

Raises
NotFoundException when no resource can be found.

get_task_schema()

Get image task schema

Returns
One Schema

Raises
NotFoundException when no resource can be found.

`get_metadef_namespace_schema()`

Get metadata definition namespace schema

Returns
One *MetadefSchema*

Raises
NotFoundException when no resource can be found.

`get_metadef_namespaces_schema()`

Get metadata definition namespaces schema

Returns
One *MetadefSchema*

Raises
NotFoundException when no resource can be found.

`get_metadef_resource_type_schema()`

Get metadata definition resource type association schema

Returns
One *MetadefSchema*

Raises
NotFoundException when no resource can be found.

`get_metadef_resource_types_schema()`

Get metadata definition resource type associations schema

Returns
One *MetadefSchema*

Raises
NotFoundException when no resource can be found.

`get_metadef_object_schema()`

Get metadata definition object schema

Returns
One *MetadefSchema*

Raises
NotFoundException when no resource can be found.

`get_metadef_objects_schema()`

Get metadata definition objects schema

Returns
One *MetadefSchema*

Raises
NotFoundException when no resource can be found.

`get_metadef_property_schema()`

Get metadata definition property schema

Returns
One *MetadefSchema*

Raises
NotFoundException when no resource can be found.

`get_metadef_properties_schema()`

Get metadata definition properties schema

Returns
One *MetadefSchema*

Raises
NotFoundException when no resource can be found.

`get_metadef_tag_schema()`

Get metadata definition tag schema

Returns
One *MetadefSchema*

Raises
NotFoundException when no resource can be found.

`get_metadef_tags_schema()`

Get metadata definition tags schema

Returns
One *MetadefSchema*

Raises
NotFoundException when no resource can be found.

Service Info Discovery Operations

`class openstack.image.v2._proxy.Proxy(session, statsd_client=None, statsd_prefix=None, prometheus_counter=None, prometheus_histogram=None, influxdb_config=None, influxdb_client=None, *args, **kwargs)`

`stores(details=False, **query)`

Return a generator of supported image stores

Returns
A generator of store objects

Return type
Store

`get_import_info()`

Get a info about image constraints

Returns
One *Import*

Raises
NotFoundException when no resource can be found.

Metadef Namespace Operations

class openstack.image.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_metadef_namespace(***attrs*)

Create a new metadef namespace from attributes

Parameters

attrs (*dict*) Keyword arguments which will be used to create a *MetadefNamespace* comprised of the properties on the MetadefNamespace class.

Returns

The results of metadef namespace creation

Return type

MetadefNamespace

delete_metadef_namespace(*metadef_namespace*, *ignore_missing=True*)

Delete a metadef namespace

Parameters

- **metadef_namespace** The value can be either the name of a metadef namespace or a *MetadefNamespace* instance.
- **ignore_missing** (*bool*) When set to `False`, *NotFoundException* will be raised when the metadef namespace does not exist.

Returns

`None`

get_metadef_namespace(*metadef_namespace*)

Get a single metadef namespace

Parameters

metadef_namespace Either the name of a metadef namespace or an *MetadefNamespace* instance.

Returns

One *MetadefNamespace*

Raises

NotFoundException when no resource can be found.

metadef_namespaces(***query*)

Return a generator of metadef namespaces

Returns

A generator object of metadef namespaces

Return type

MetadefNamespace

Raises
NotFoundException when no resource can be found.

update_metadef_namespace(*metadef_namespace*, ***attrs*)

Update a server

Parameters

- **metadef_namespace** Either the name of a metadef namespace or an *MetadefNamespace* instance.
- **attrs** The attributes to update on the metadef namespace represented by `metadef_namespace`.

Returns
The updated metadef namespace

Return type
MetadefNamespace

Metadef Object Operations

class openstack.image.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_metadef_object(*namespace*, ***attrs*)

Create a new object from namespace

Parameters

- **namespace** The value can be either the name of a metadef namespace or a *MetadefNamespace* instance.
- **attrs** (*dict*) Keyword arguments which will be used to create a *MetadefObject*, comprised of the properties on the Metadef object class.

Returns
A metadef namespace

Return type
MetadefObject

get_metadef_object(*metadef_object*, *namespace*)

Get a single metadef object

Parameters

- **metadef_object** The value can be the ID of a metadef_object or a *MetadefObject* instance.
- **namespace** The value can be either the name of a metadef namespace or a *MetadefNamespace* instance.

Returns
One *MetadefObject*

Raises
NotFoundException when no resource can be found.

metadef_objects(*namespace*)

Get metadef object list of the namespace

Parameters
namespace The value can be either the name of a metadef namespace or a *MetadefNamespace* instance.

Returns
One *MetadefObject*

Raises
NotFoundException when no resource can be found.

update_metadef_object(*metadef_object*, *namespace*, ***attrs*)

Update a single metadef object

Parameters

- **metadef_object** The value can be the ID of a metadef_object or a *MetadefObject* instance.
- **namespace** The value can be either the name of a metadef namespace or a *MetadefNamespace* instance.
- **attrs** (*dict*) Keyword arguments which will be used to update a *MetadefObject*

Returns
One *MetadefObject*

Raises
NotFoundException when no resource can be found.

delete_metadef_object(*metadef_object*, *namespace*, ***attrs*)

Removes a single metadef object

Parameters

- **metadef_object** The value can be the ID of a metadef_object or a *MetadefObject* instance.
- **namespace** The value can be either the name of a metadef namespace or a *MetadefNamespace* instance.
- **attrs** (*dict*) Keyword arguments which will be used to update a *MetadefObject*

Returns
None

Raises
NotFoundException when no resource can be found.

Metadef Resource Type Operations

class openstack.image.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

metadef_resource_types(***query*)

Return a generator of metadef resource types

Returns
A generator object of metadef resource types

Return type
MetadefResourceType

Raises
NotFoundException when no resource can be found.

create_metadef_resource_type_association(*metadef_namespace*, ***attrs*)

Creates a resource type association between a namespace
and the resource type specified in the body of the request.

Parameters
attrs (*dict*) Keyword arguments which will be used to create a *MetadefResourceTypeAssociation* comprised of the properties on the MetadefResourceTypeAssociation class.

Returns
The results of metadef resource type association creation

Return type
MetadefResourceTypeAssociation

delete_metadef_resource_type_association(*metadef_resource_type*, *metadef_namespace*, *ignore_missing=True*)

Removes a resource type association in a namespace.

Parameters

- **metadef_resource_type** The value can be either the name of a metadef resource type association or an *MetadefResourceTypeAssociation* instance.
- **metadef_namespace** The value can be either the name of metadef namespace or an *MetadefNamespace* instance
- **ignore_missing** (*bool*) When set to `False`, *NotFoundException* will be raised when the metadef resource type association does not exist.

Returns
`None`

metadef_resource_type_associations(*metadef_namespace*, ***query*)

Return a generator of metadef resource type associations

Parameters
metadef_namespace The value can be either the name of metadef namespace or an *MetadefNamespace* instance

Returns
A generator object of metadef resource type associations

Return type
MetadefResourceTypeAssociation

Raises
NotFoundException when no resource can be found.

Metadef Property Operations

class openstack.image.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_metadef_property(*metadef_namespace*, ***attrs*)

Create a metadef property

Parameters

- **metadef_namespace** The value can be either the name of metadef namespace or an `MetadefNamespace` instance
- **attrs** The attributes to create on the metadef property represented by `metadef_property`.

Returns
The created metadef property

Return type
MetadefProperty

update_metadef_property(*metadef_property*, *metadef_namespace*, ***attrs*)

Update a metadef property

Parameters

- **metadef_property** The value can be either the name of metadef property or an *MetadefProperty* instance.
- **metadef_namespace** The value can be either the name of metadef namespace or an *MetadefNamespace* instance
- **attrs** The attributes to update on the metadef property represented by `metadef_property`.

Returns
The updated metadef property

Return type
MetadefProperty

delete_metadef_property(*metadef_property*, *metadef_namespace*, *ignore_missing=True*)

Delete a metadef property

Parameters

- **metadef_property** The value can be either the name of metadef property or an *MetadefProperty* instance
- **metadef_namespace** The value can be either the name of metadef namespace or an *MetadefNamespace* instance
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the instance does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent instance.

Returns

None

get_metadef_property(*metadef_property*, *metadef_namespace*, ***query*)

Get a single metadef property

Parameters

- **metadef_property** The value can be either the name of metadef property or an *MetadefProperty* instance.
- **metadef_namespace** The value can be either the name of metadef namespace or an *MetadefNamespace* instance

Returns

One *MetadefProperty*

Raises

NotFoundException when no resource can be found.

Helpers

class openstack.image.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

wait_for_delete(*res*, *interval=2*, *wait=120*, *callback=None*)

Wait for a resource to be deleted.

Parameters

- **res** The resource to wait on to be deleted.
- **interval** Number of seconds to wait before to consecutive checks.
- **wait** Maximum number of seconds to wait before the change.
- **callback** A callback function. This will be called with a single value, progress, which is a percentage value from 0-100.

Returns

The resource is returned on success.

Raises

ResourceTimeout if transition to delete failed to occur in the specified seconds.

Cache Operations

class openstack.image.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

cache_delete_image(*image*, *ignore_missing=True*)

Delete an image from cache.

Parameters

- **image** The value can be either the name of an image or a *Image* instance.
- **ignore_missing** (*bool*) When set to `False`, *NotFoundException* will be raised when the metadef namespace does not exist.

Returns

`None`

queue_image(*image_id*)

Queue image(s) for caching.

clear_cache(*target='both'*)

Clear all images from cache, queue or both

Parameters

target Specify which target you want to clear One of: `both``(default),` `cache`, `queue`.

KeyManager API

For details on how to use key_management, see *Using OpenStack Key Manager*

The KeyManager Class

The key_management high-level interface is available through the `key_manager` member of a *Connection* object. The `key_manager` member will only be added if the service is detected.

Secret Operations

class openstack.key_manager.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_secret(***attrs*)

Create a new secret from attributes

Parameters

attrs (*dict*) Keyword arguments which will be used to create a *Secret*, comprised of the properties on the Order class.

Returns

The results of secret creation

Return type

Secret

delete_secret(*secret*, *ignore_missing=True*)

Delete a secret

Parameters

- **secret** The value can be either the ID of a secret or a *Secret* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the secret does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent secret.

Returns

`None`

find_secret(*name_or_id*, *ignore_missing=True*)

Find a single secret

Parameters

- **name_or_id** The name or ID of a secret.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns

One *Secret* or None

get_secret(*secret*)

Get a single secret

Parameters

secret The value can be the ID of a secret or a *Secret* instance.

Returns

One *Secret*

Raises

NotFoundException when no resource can be found.

secrets(***query*)

Return a generator of secrets

Parameters

query (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of secret objects

Return type
Secret

`update_secret`(*secret*, ***attrs*)

Update a secret

Parameters

- **secret** Either the id of a secret or a *Secret* instance.
- **attrs** The attributes to update on the secret represented by `secret`.

Returns
The updated secret

Return type
Secret

Container Operations

`class openstack.key_manager.v1._proxy.Proxy`(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

`create_container`(***attrs*)

Create a new container from attributes

Parameters
attrs (*dict*) Keyword arguments which will be used to create a *Container*, comprised of the properties on the Container class.

Returns
The results of container creation

Return type
Container

`delete_container`(*container*, *ignore_missing=True*)

Delete a container

Parameters

- **container** The value can be either the ID of a container or a *Container* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the container does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent container.

Returns
`None`

`find_container`(*name_or_id*, *ignore_missing=True*)

Find a single container

Parameters

- **name_or_id** The name or ID of a container.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns

One *Container* or None

`get_container`(*container*)

Get a single container

Parameters

container The value can be the ID of a container or a *Container* instance.

Returns

One *Container*

Raises

NotFoundException when no resource can be found.

`containers`(***query*)

Return a generator of containers

Parameters

query (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns

A generator of container objects

Return type

Container

`update_container`(*container*, ***attrs*)

Update a container

Parameters

- **container** Either the id of a container or a *Container* instance.
- **attrs** The attributes to update on the container represented by `container`.

Returns

The updated container

Return type

Container

Order Operations

class openstack.key_manager.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_order(***attrs*)

Create a new order from attributes

Parameters

attrs (*dict*) Keyword arguments which will be used to create a *Order*, comprised of the properties on the Order class.

Returns

The results of order creation

Return type

Order

delete_order(*order*, *ignore_missing=True*)

Delete an order

Parameters

- **order** The value can be either the ID of a order or a *Order* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the order does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent order.

Returns

`None`

find_order(*name_or_id*, *ignore_missing=True*)

Find a single order

Parameters

- **name_or_id** The name or ID of a order.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns

One *Order* or None

get_order(*order*)

Get a single order

Parameters

order The value can be the ID of an order or a *Order* instance.

Returns

One *Order*

Raises

NotFoundException when no resource can be found.

orders(***query*)

Return a generator of orders

Parameters

query (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns

A generator of order objects

Return type

Order

update_order(*order*, ***attrs*)

Update a order

Parameters

- **order** Either the id of a order or a *Order* instance.
- **attrs** The attributes to update on the order represented by order.

Returns

The updated order

Return type

Order

Load Balancer v2 API

The LoadBalancer Class

The load_balancer high-level interface is available through the load_balancer member of a *Connection* object. The load_balancer member will only be added if the service is detected.

Load Balancer Operations

class openstack.load_balancer.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_load_balancer(***attrs*)

Create a new load balancer from attributes

Parameters

attrs (*dict*) Keyword arguments which will be used to create a *LoadBalancer*, comprised of the properties on the LoadBalancer class.

Returns

The results of load balancer creation

Return type

LoadBalancer

get_load_balancer(**attrs*)

Get a load balancer

Parameters
load_balancer The value can be the ID of a load balancer or *LoadBalancer* instance.

Returns
One *LoadBalancer*

get_load_balancer_statistics(*load_balancer*)

Get the load balancer statistics

Parameters
load_balancer The value can be the ID of a load balancer or *LoadBalancer* instance.

Returns
One *LoadBalancerStats*

load_balancers(***query*)

Retrieve a generator of load balancers

Returns
A generator of load balancer instances

delete_load_balancer(*load_balancer*, *ignore_missing=True*, *cascade=False*)

Delete a load balancer

Parameters

- **load_balancer** The load_balancer can be either the ID or a *LoadBalancer* instance
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the load balancer does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent load balancer.
- **cascade** (*bool*) If true will delete all child objects of the load balancer.

Returns
`None`

find_load_balancer(*name_or_id*, *ignore_missing=True*)

Find a single load balancer

Parameters

- **name_or_id** The name or ID of a load balancer
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the load balancer does not exist. When set to `True`, no exception will be set when attempting to find a nonexistent load balancer.

Returns
`None`

update_load_balancer(*load_balancer*, ***attrs*)

Update a load balancer

Parameters

- **load_balancer** The load_balancer can be either the ID or a *LoadBalancer* instance
- **attrs** (*dict*) The attributes to update on the load balancer represented by load_balancer.

Returns
The updated load_balancer

Return type
LoadBalancer

wait_for_load_balancer(*name_or_id*, *status='ACTIVE'*, *failures=['ERROR']*, *interval=2*, *wait=300*)

Wait for load balancer status

Parameters

- **name_or_id** The name or ID of the load balancer.
- **status** Desired status.
- **failures** (list) Statuses that would be interpreted as failures. Default to [ERROR].
- **interval** Number of seconds to wait between consecutive checks. Defaults to 2.
- **wait** Maximum number of seconds to wait before the status to be reached. Defaults to 300.

Returns
The load balancer is returned on success.

Raises
ResourceTimeout if transition to the desired status failed to occur within the specified wait time.

Raises
ResourceFailure if the resource has transited to one of the failure statuses.

Raises
AttributeError if the resource does not have a status attribute.

failover_load_balancer(*load_balancer*)

Failover a load balancer

Parameters
load_balancer The value can be the ID of a load balancer or *LoadBalancer* instance.

Returns
None

Listener Operations

class openstack.load_balancer.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_listener(***attrs*)

Create a new listener from attributes

Parameters

attrs (*dict*) Keyword arguments which will be used to create a *Listener*, comprised of the properties on the Listener class.

Returns

The results of listener creation

Return type

Listener

delete_listener(*listener*, *ignore_missing=True*)

Delete a listener

Parameters

- **listener** The value can be either the ID of a listener or a *Listener* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the listner does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent listener.

Returns

`None`

find_listener(*name_or_id*, *ignore_missing=True*)

Find a single listener

Parameters

- **name_or_id** The name or ID of a listener.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns

One *Listener* or None

get_listener(*listener*)

Get a single listener

Parameters

listener The value can be the ID of a listener or a *Listener* instance.

Returns

One *Listener*

Raises
NotFoundException when no resource can be found.

`get_listener_statistics`(*listener*)
Get the listener statistics

Parameters
listener The value can be the ID of a listener or a *Listener* instance.

Returns
One *ListenerStats*

Raises
NotFoundException when no resource can be found.

`listeners`(***query*)
Return a generator of listeners

Parameters
query (*dict*) Optional query parameters to be sent to limit the resources being returned. Valid parameters are:

Returns
A generator of listener objects

Return type
Listener

`update_listener`(*listener*, ***attrs*)
Update a listener

Parameters

- **listener** Either the id of a listener or a *Listener* instance.
- **attrs** (*dict*) The attributes to update on the listener represented by `listener`.

Returns
The updated listener

Return type
Listener

Pool Operations

`class openstack.load_balancer.v2._proxy.Proxy`(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

`create_pool`(***attrs*)
Create a new pool from attributes

Parameters
attrs (*dict*) Keyword arguments which will be used to create a *Pool*, comprised of the properties on the Pool class.

Returns
The results of Pool creation

Return type
Pool

`get_pool`(**attrs*)

Get a pool

Parameters
pool Value is either a pool ID or a *Pool* instance.

Returns
One *Pool*

`pools`(***query*)

Retrieve a generator of pools

Returns
A generator of Pool instances

`delete_pool`(*pool*, *ignore_missing=True*)

Delete a pool

Parameters

- **pool** The pool is either a pool ID or a *Pool* instance
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the pool does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent pool.

Returns
`None`

`find_pool`(*name_or_id*, *ignore_missing=True*)

Find a single pool

Parameters

- **name_or_id** The name or ID of a pool
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the pool does not exist. When set to `True`, no exception will be set when attempting to find a nonexistent pool.

Returns
`None`

`update_pool`(*pool*, ***attrs*)

Update a pool

Parameters

- **pool** Either the id of a pool or a *Pool* instance.
- **attrs** (*dict*) The attributes to update on the pool represented by `pool`.

Returns
The updated pool

Return type
Pool

Member Operations

class openstack.load_balancer.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_member(*pool*, ***attrs*)

Create a new member from attributes

Parameters

- **pool** The pool can be either the ID of a pool or a *Pool* instance that the member will be created in.
- **attrs** (*dict*) Keyword arguments which will be used to create a *Member*, comprised of the properties on the Member class.

Returns
The results of member creation

Return type
Member

delete_member(*member*, *pool*, *ignore_missing=True*)

Delete a member

Parameters

- **member** The member can be either the ID of a member or a *Member* instance.
- **pool** The pool can be either the ID of a pool or a *Pool* instance that the member belongs to.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the member does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent member.

Returns
`None`

find_member(*name_or_id*, *pool*, *ignore_missing=True*)

Find a single member

Parameters

- **name_or_id** (*str*) The name or ID of a member.
- **pool** The pool can be either the ID of a pool or a *Pool* instance that the member belongs to.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns
One *Member* or None

get_member(*member*, *pool*)

Get a single member

Parameters

- **member** The member can be the ID of a member or a *Member* instance.
- **pool** The pool can be either the ID of a pool or a *Pool* instance that the member belongs to.

Returns
One *Member*

Raises
NotFoundException when no resource can be found.

members(*pool*, ***query*)

Return a generator of members

Parameters

- **pool** The pool can be either the ID of a pool or a *Pool* instance that the member belongs to.
- **query** (*dict*) Optional query parameters to be sent to limit the resources being returned. Valid parameters are:

Returns
A generator of member objects

Return type
Member

update_member(*member*, *pool*, ***attrs*)

Update a member

Parameters

- **member** Either the ID of a member or a *Member* instance.
- **pool** The pool can be either the ID of a pool or a *Pool* instance that the member belongs to.
- **attrs** (*dict*) The attributes to update on the member represented by `member`.

Returns
The updated member

Return type
Member

Health Monitor Operations

class openstack.load_balancer.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

find_health_monitor(*name_or_id*, *ignore_missing=True*)

Find a single health monitor

Parameters

- **name_or_id** The name or ID of a health monitor
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the health monitor does not exist. When set to `True`, no exception will be set when attempting to find a nonexistent health monitor.

Returns

The `openstack.load_balancer.v2.healthmonitor.HealthMonitor` object matching the given name or id or None if nothing matches.

Raises

openstack.exceptions.DuplicateResource if more than one resource is found for this request.

Raises

openstack.exceptions.NotFoundException if nothing is found and ignore_missing is `False`.

create_health_monitor(***attrs*)

Create a new health monitor from attributes

Parameters

attrs (*dict*) Keyword arguments which will be used to create a `HealthMonitor`, comprised of the properties on the HealthMonitor class.

Returns

The results of HealthMonitor creation

Return type

`HealthMonitor`

get_health_monitor(*healthmonitor*)

Get a health monitor

Parameters

healthmonitor The value can be the ID of a health monitor or `HealthMonitor` instance.

Returns

One health monitor

Return type

`HealthMonitor`

health_monitors(***query*)

Retrieve a generator of health monitors

Parameters

query (*dict*) Optional query parameters to be sent to limit the resources being returned. Valid parameters are: name, created_at, updated_at, delay, expected_codes, http_method, max_retries, max_retries_down, pool_id, provisioning_status, operating_status, timeout, project_id, type, url_path, is_admin_state_up.

Returns

A generator of health monitor instances

delete_health_monitor(*healthmonitor*, *ignore_missing=True*)

Delete a health monitor

Parameters

- **healthmonitor** The healthmonitor can be either the ID of the health monitor or a `HealthMonitor` instance
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the healthmonitor does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent healthmonitor.

Returns

`None`

update_health_monitor(*healthmonitor*, ***attrs*)

Update a health monitor

Parameters

- **healthmonitor** The healthmonitor can be either the ID of the health monitor or a `HealthMonitor` instance
- **attrs** (*dict*) The attributes to update on the health monitor represented by `healthmonitor`.

Returns

The updated health monitor

Return type

`HealthMonitor`

L7 Policy Operations

class openstack.load_balancer.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_l7_policy(***attrs*)

Create a new l7policy from attributes

Parameters

attrs (*dict*) Keyword arguments which will be used to create a *L7Policy*, comprised of the properties on the L7Policy class.

Returns
The results of l7policy creation

Return type
L7Policy

delete_l7_policy(*l7_policy*, *ignore_missing=True*)

Delete a l7policy

Parameters

- **l7_policy** The value can be either the ID of a l7policy or a *L7Policy* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the l7policy does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent l7policy.

Returns
`None`

find_l7_policy(*name_or_id*, *ignore_missing=True*)

Find a single l7policy

Parameters

- **name_or_id** The name or ID of a l7policy.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns
One *L7Policy* or None

get_l7_policy(*l7_policy*)

Get a single l7policy

Parameters
l7_policy The value can be the ID of a l7policy or a *L7Policy* instance.

Returns
One *L7Policy*

Raises
NotFoundException when no resource can be found.

l7_policies(***query*)

Return a generator of l7policies

Parameters
query (*dict*) Optional query parameters to be sent to limit the resources being returned. Valid parameters are:

Returns
A generator of l7policy objects

Return type
L7Policy

update_l7_policy(*l7_policy*, ***attrs*)

Update a l7policy

Parameters

- **l7_policy** Either the id of a l7policy or a *L7Policy* instance.
- **attrs** (*dict*) The attributes to update on the l7policy represented by `l7policy`.

Returns

The updated l7policy

Return type

L7Policy

L7 Rule Operations

class openstack.load_balancer.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_l7_rule(*l7_policy*, ***attrs*)

Create a new l7rule from attributes

Parameters

- **l7_policy** The l7_policy can be either the ID of a l7policy or *L7Policy* instance that the l7rule will be created in.
- **attrs** (*dict*) Keyword arguments which will be used to create a *L7Rule*, comprised of the properties on the L7Rule class.

Returns

The results of l7rule creation

Return type

L7Rule

delete_l7_rule(*l7rule*, *l7_policy*, *ignore_missing=True*)

Delete a l7rule

Parameters

- **l7rule** The l7rule can be either the ID of a l7rule or a *L7Rule* instance.
- **l7_policy** The l7_policy can be either the ID of a l7policy or *L7Policy* instance that the l7rule belongs to.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the l7rule does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent l7rule.

Returns

`None`

find_l7_rule(*name_or_id*, *l7_policy*, *ignore_missing=True*)

Find a single l7rule

Parameters

- **name_or_id** (*str*) The name or ID of a l7rule.
- **l7_policy** The l7_policy can be either the ID of a l7policy or *L7Policy* instance that the l7rule belongs to.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns

One *L7Rule* or None

get_l7_rule(*l7rule*, *l7_policy*)

Get a single l7rule

Parameters

- **l7rule** The l7rule can be the ID of a l7rule or a *L7Rule* instance.
- **l7_policy** The l7_policy can be either the ID of a l7policy or *L7Policy* instance that the l7rule belongs to.

Returns

One *L7Rule*

Raises

NotFoundException when no resource can be found.

l7_rules(*l7_policy*, ***query*)

Return a generator of l7rules

Parameters

- **l7_policy** The l7_policy can be either the ID of a l7_policy or *L7Policy* instance that the l7rule belongs to.
- **query** (*dict*) Optional query parameters to be sent to limit the resources being returned. Valid parameters are:

Returns

A generator of l7rule objects

Return type

L7Rule

update_l7_rule(*l7rule*, *l7_policy*, ***attrs*)

Update a l7rule

Parameters

- **l7rule** Either the ID of a l7rule or a *L7Rule* instance.
- **l7_policy** The l7_policy can be either the ID of a l7policy or *L7Policy* instance that the l7rule belongs to.
- **attrs** (*dict*) The attributes to update on the l7rule represented by `l7rule`.

Returns
The updated l7rule

Return type
L7Rule

Provider Operations

class openstack.load_balancer.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

providers(***query*)

Retrieve a generator of providers

Returns
A generator of providers instances

provider_flavor_capabilities(*provider*, ***query*)

Retrieve a generator of provider flavor capabilities

Returns
A generator of provider flavor capabilities instances

Flavor Profile Operations

class openstack.load_balancer.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_flavor_profile(***attrs*)

Create a new flavor profile from attributes

Parameters
attrs (*dict*) Keyword arguments which will be used to create a *FlavorProfile*, comprised of the properties on the FlavorProfile class.

Returns
The results of profile creation creation

Return type
FlavorProfile

get_flavor_profile(**attrs*)

Get a flavor profile

Parameters
flavor_profile The value can be the name of a flavor profile or *FlavorProfile* instance.

Returns
One *FlavorProfile*

flavor_profiles(***query*)

Retrieve a generator of flavor profiles

Returns
A generator of flavor profiles instances

delete_flavor_profile(*flavor_profile*, *ignore_missing=True*)

Delete a flavor profile

Parameters

- **flavor_profile** The flavor_profile can be either the ID or a *FlavorProfile* instance
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the flavor profile does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent flavor profile.

Returns
`None`

find_flavor_profile(*name_or_id*, *ignore_missing=True*)

Find a single flavor profile

Parameters

- **name_or_id** The name or ID of a flavor profile
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the flavor profile does not exist. When set to `True`, no exception will be set when attempting to find a nonexistent flavor profile.

Returns
`None`

update_flavor_profile(*flavor_profile*, ***attrs*)

Update a flavor profile

Parameters

- **flavor_profile** The flavor_profile can be either the ID or a *FlavorProfile* instance
- **attrs** (*dict*) The attributes to update on the flavor profile represented by `flavor_profile`.

Returns
The updated flavor profile

Return type
FlavorProfile

Flavor Operations

class openstack.load_balancer.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_flavor(***attrs*)

Create a new flavor from attributes

Parameters

attrs (*dict*) Keyword arguments which will be used to create a *Flavor*, comprised of the properties on the Flavorclass.

Returns

The results of flavor creation creation

Return type

Flavor

get_flavor(**attrs*)

Get a flavor

Parameters

flavor The value can be the ID of a flavor or *Flavor* instance.

Returns

One *Flavor*

flavors(***query*)

Retrieve a generator of flavors

Returns

A generator of flavor instances

delete_flavor(*flavor*, *ignore_missing=True*)

Delete a flavor

Parameters

- **flavor** The flavorcan be either the ID or a *Flavor* instance
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the flavor does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent flavor.

Returns

`None`

find_flavor(*name_or_id*, *ignore_missing=True*)

Find a single flavor

Parameters

- **name_or_id** The name or ID of a flavor

- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the flavor does not exist. When set to `True`, no exception will be set when attempting to find a nonexistent flavor.

Returns
None

update_flavor(*flavor*, ***attrs*)

Update a flavor

Parameters

- **flavor** The flavor can be either the ID or a *Flavor* instance
- **attrs** (*dict*) The attributes to update on the flavor represented by `flavor`.

Returns
The updated flavor

Return type
Flavor

Quota Operations

class openstack.load_balancer.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

quotas(***query*)

Return a generator of quotas

Parameters
query (*dict*) Optional query parameters to be sent to limit the resources being returned. Currently no query parameter is supported.

Returns
A generator of quota objects

Return type
Quota

get_quota(*quota*)

Get a quota

Parameters
quota The value can be the ID of a quota or a *Quota* instance. The ID of a quota is the same as the project ID for the quota.

Returns
One *Quota*

Raises
NotFoundException when no resource can be found.

update_quota(*quota*, ***attrs*)

Update a quota

Parameters

- **quota** Either the ID of a quota or a *Quota* instance. The ID of a quota is the same as the project ID for the quota.
- **attrs** (*dict*) The attributes to update on the quota represented by `quota`.

Returns
The updated quota

Return type
Quota

get_quota_default()

Get a default quota

Returns
One `QuotaDefault`

delete_quota(*quota*, *ignore_missing=True*)

Delete a quota (i.e. reset to the default quota)

Parameters

- **quota** The value can be either the ID of a quota or a *Quota* instance. The ID of a quota is the same as the project ID for the quota.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when quota does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent quota.

Returns
`None`

Amphora Operations

class openstack.load_balancer.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

amphorae(***query*)

Retrieve a generator of amphorae

Returns
A generator of amphora instances

get_amphora(**attrs*)

Get a amphora

Parameters
amphora The value can be the ID of an amphora or *Amphora* instance.

Returns
One *Amphora*

find_amphora(*amphora_id*, *ignore_missing=True*)

Find a single amphora

Parameters

- **amphora_id** The ID of a amphora
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the amphora does not exist. When set to `True`, no exception will be set when attempting to find a nonexistent amphora.

Returns
`None`

configure_amphora(*amphora_id*)

Update the configuration of an amphora agent

Parameters
amphora_id The ID of an amphora

Returns
`None`

failover_amphora(*amphora_id*)

Failover an amphora

Parameters
amphora_id The ID of an amphora

Returns
`None`

Availability Zone Profile Operations

class openstack.load_balancer.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_availability_zone_profile(***attrs*)

Create a new availability zone profile from attributes

Parameters
attrs (*dict*) Keyword arguments which will be used to create a *AvailabilityZoneProfile* comprised of the properties on the AvailabilityZoneProfile class.

Returns
The results of profile creation

Return type
AvailabilityZoneProfile

get_availability_zone_profile(**attrs*)

Get an availability zone profile

Parameters

availability_zone_profile The value can be the ID of an availability_zone profile or *AvailabilityZoneProfile* instance.

Returns

One *AvailabilityZoneProfile*

availability_zone_profiles(***query*)

Retrieve a generator of availability zone profiles

Returns

A generator of availability zone profiles instances

delete_availability_zone_profile(*availability_zone_profile*, *ignore_missing=True*)

Delete an availability zone profile

Parameters

- **availability_zone_profile** The availability_zone_profile can be either the ID or a *AvailabilityZoneProfile* instance
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the availability zone profile does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent availability zone profile.

Returns

`None`

find_availability_zone_profile(*name_or_id*, *ignore_missing=True*)

Find a single availability zone profile

Parameters

- **name_or_id** The name or ID of a availability zone profile
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the availability zone profile does not exist. When set to `True`, no exception will be set when attempting to find a nonexistent availability zone profile.

Returns

`None`

update_availability_zone_profile(*availability_zone_profile*, ***attrs*)

Update an availability zone profile

Parameters

- **availability_zone_profile** The availability_zone_profile can be either the ID or a *AvailabilityZoneProfile* instance
- **attrs** (*dict*) The attributes to update on the availability_zone profile represented by `availability_zone_profile`.

Returns

The updated availability zone profile

Return type
AvailabilityZoneProfile

Availability Zone Operations

class openstack.load_balancer.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_availability_zone(***attrs*)

Create a new availability zone from attributes

Parameters
attrs (*dict*) Keyword arguments which will be used to create a *AvailabilityZone* comprised of the properties on the AvailabilityZoneclass.

Returns
The results of availability_zone creation creation

Return type
AvailabilityZone

get_availability_zone(**attrs*)

Get an availability zone

Parameters
availability_zone The value can be the ID of a availability_zone or *AvailabilityZone* instance.

Returns
One *AvailabilityZone*

availability_zones(***query*)

Retrieve a generator of availability zones

Returns
A generator of availability zone instances

delete_availability_zone(*availability_zone*, *ignore_missing=True*)

Delete an availability_zone

Parameters

- **availability_zone** The availability_zone can be either the ID or a *AvailabilityZone* instance
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the availability zone does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent availability zone.

Returns
`None`

find_availability_zone(*name_or_id*, *ignore_missing=True*)

Find a single availability zone

Parameters

- **name_or_id** The name or ID of a availability zone
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the availability zone does not exist. When set to `True`, no exception will be set when attempting to find a nonexistent availability zone.

Returns
`None`

update_availability_zone(*availability_zone*, ***attrs*)

Update an availability zone

Parameters

- **availability_zone** The availability_zone can be either the ID or a *AvailabilityZone* instance
- **attrs** (*dict*) The attributes to update on the availability_zone represented by `availability_zone`.

Returns
The updated availability_zone

Return type
AvailabilityZone

Message API v2

For details on how to use message, see *Using OpenStack Message*

The Message v2 Class

The message high-level interface is available through the `message` member of a *Connection* object. The `message` member will only be added if the service is detected.

Message Operations

class openstack.message.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

post_message(*queue_name*, *messages*)

Post messages to given queue

Parameters

- **queue_name** The name of target queue to post message to.
- **messages** List of messages body and TTL to post. :type messages: `list`

Returns
A string includes location of messages successfully posted.

`messages`(*queue_name*, ***query*)

Retrieve a generator of messages

Parameters

- **queue_name** The name of target queue to query messages from.
- **query** (*kwargs*) Optional query parameters to be sent to restrict the messages to be returned. Available parameters include:
  - **limit: Requests at most the specified number of items be** returned from the query.
  - **marker: Specifies the ID of the last-seen subscription. Use the** limit parameter to make an initial limited request and use the ID of the last-seen subscription from the response as the marker parameter value in a subsequent limited request.
  - **echo: Indicate if the messages can be echoed back to the client** that posted them.
  - **include_claimed: Indicate if the messages list should include** the claimed messages.

Returns
A generator of message instances.

`get_message`(*queue_name*, *message*)

Get a message

Parameters

- **queue_name** The name of target queue to get message from.
- **message** The value can be the name of a message or a `Message` instance.

Returns
One `Message`

Raises
NotFoundException when no message matching the criteria could be found.

`delete_message`(*queue_name*, *value*, *claim=None*, *ignore_missing=True*)

Delete a message

Parameters

- **queue_name** The name of target queue to delete message from.
- **value** The value can be either the name of a message or a `Message` instance.
- **claim** The value can be the ID or a `Claim` instance of the claim seizing the message. If None, the message has not been claimed.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the message does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent message.

Returns
None

Queue Operations

class openstack.message.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_queue(***attrs*)

Create a new queue from attributes

Parameters
attrs (*dict*) Keyword arguments which will be used to create a Queue, comprised of the properties on the Queue class.

Returns
The results of queue creation

Return type
Queue

get_queue(*queue*)

Get a queue

Parameters
queue The value can be the name of a queue or a Queue instance.

Returns
One Queue

Raises
NotFoundException when no queue matching the name could be found.

queues(***query*)

Retrieve a generator of queues

Parameters
query (*kwargs*) Optional query parameters to be sent to restrict the queues to be returned. Available parameters include:

- **limit: Requests at most the specified number of items be** returned from the query.
- **marker: Specifies the ID of the last-seen queue. Use the limit** parameter to make an initial limited request and use the ID of the last-seen queue from the response as the marker parameter value in a subsequent limited request.

Returns
A generator of queue instances.

delete_queue(*value*, *ignore_missing=True*)

Delete a queue

Parameters

- **value** The value can be either the name of a queue or a `Queue` instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the queue does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent queue.

Returns
`None`

Claim Operations

class openstack.message.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_claim(*queue_name*, ***attrs*)

Create a new claim from attributes

Parameters

- **queue_name** The name of target queue to claim message from.
- **attrs** (*dict*) Keyword arguments which will be used to create a `Claim`, comprised of the properties on the Claim class.

Returns
The results of claim creation

Return type
`Claim`

get_claim(*queue_name*, *claim*)

Get a claim

Parameters

- **queue_name** The name of target queue to claim message from.
- **claim** The value can be either the ID of a claim or a `Claim` instance.

Returns
One `Claim`

Raises
NotFoundException when no claim matching the criteria could be found.

update_claim(*queue_name*, *claim*, ***attrs*)

Update an existing claim from attributes

Parameters

- **queue_name** The name of target queue to claim message from.
- **claim** The value can be either the ID of a claim or a `Claim` instance.
- **attrs** (*dict*) Keyword arguments which will be used to update a `Claim`, comprised of the properties on the Claim class.

Returns
The results of claim update

Return type
`Claim`

`delete_claim`(*queue_name*, *claim*, *ignore_missing=True*)

Delete a claim

Parameters

- **queue_name** The name of target queue to claim messages from.
- **claim** The value can be either the ID of a claim or a `Claim` instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the claim does not exist. When set to `True`, no exception will be thrown when attempting to delete a nonexistent claim.

Returns
`None`

Subscription Operations

`class openstack.message.v2._proxy.Proxy`(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

`create_subscription`(*queue_name*, ***attrs*)

Create a new subscription from attributes

Parameters

- **queue_name** The name of target queue to subscribe on.
- **attrs** (*dict*) Keyword arguments which will be used to create a `Subscription`, comprised of the properties on the Subscription class.

Returns
The results of subscription creation

Return type
`Subscription`

`subscriptions`(*queue_name*, ***query*)

Retrieve a generator of subscriptions

Parameters

- **queue_name** The name of target queue to subscribe on.
- **query** (*kwargs*) Optional query parameters to be sent to restrict the subscriptions to be returned. Available parameters include:
  - **limit: Requests at most the specified number of items be** returned from the query.

- **marker: Specifies the ID of the last-seen subscription. Use the** limit parameter to make an initial limited request and use the ID of the last-seen subscription from the response as the marker parameter value in a subsequent limited request.

Returns
A generator of subscription instances.

`get_subscription`(*queue_name*, *subscription*)

Get a subscription

Parameters

- **queue_name** The name of target queue of subscription.
- **message** The value can be the ID of a subscription or a `Subscription` instance.

Returns
One `Subscription`

Raises
NotFoundException when no subscription matching the criteria could be found.

`delete_subscription`(*queue_name*, *value*, *ignore_missing=True*)

Delete a subscription

Parameters

- **queue_name** The name of target queue to delete subscription from.
- **value** The value can be either the name of a subscription or a `Subscription` instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the subscription does not exist. When set to `True`, no exception will be thrown when attempting to delete a nonexistent subscription.

Returns
`None`

Network API

For details on how to use network, see *Using OpenStack Network*

The Network Class

The network high-level interface is available through the `network` member of a *Connection* object. The `network` member will only be added if the service is detected.

Network Operations

`class openstack.network.v2._proxy.Proxy`(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

dhcp_agent_hosting_networks(*agent*, ***query*)

A generator of networks hosted by a DHCP agent.

Parameters

- **agent** Either the agent id of an instance of `Agent`
- **query** kwargs query: Optional query parameters to be sent to limit the resources being returned.

Returns

A generator of networks

add_dhcp_agent_to_network(*agent*, *network*)

Add a DHCP Agent to a network

Parameters

- **agent** Either the agent id of an instance of `Agent`
- **network** Network instance

Returns

remove_dhcp_agent_from_network(*agent*, *network*)

Remove a DHCP Agent from a network

Parameters

- **agent** Either the agent id of an instance of `Agent`
- **network** Network instance

Returns

create_network(***attrs*)

Create a new network from attributes

Parameters

attrs Keyword arguments which will be used to create a *Network*, comprised of the properties on the Network class.

Returns

The results of network creation

Return type

Network

delete_network(*network*, *ignore_missing=True*, *if_revision=None*)

Delete a network

Parameters

- **network** The value can be either the ID of a network or a *Network* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the network does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent network.

- **if_revision** (*int*) Revision to put in If-Match header of update request to perform compare-and-swap update.

Returns
None

find_network(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single network

Parameters

- **name_or_id** The name or ID of a network.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns
One *Network* or None

get_network(*network*)

Get a single network

Parameters
network The value can be the ID of a network or a *Network* instance.

Returns
One *Network*

Raises
NotFoundException when no resource can be found.

networks(***query*)

Return a generator of networks

Parameters
query (*kwargs*) Optional query parameters to be sent to limit the resources being returned. Available parameters include:

- `description`: The network description.
- `ipv4_address_scope_id`: The ID of the IPv4 address scope for the network.
- `ipv6_address_scope_id`: The ID of the IPv6 address scope for the network.
- `is_admin_state_up`: Network administrative state
- `is_port_security_enabled`: The port security status.
- `is_router_external`: Network is external or not.
- `is_shared`: Whether the network is shared across projects.
- `name`: The name of the network.
- `status`: Network status

- `project_id`: Owner tenant ID
- `provider_network_type`: Network physical mechanism
- `provider_physical_network`: Physical network
- `provider_segmentation_id`: VLAN ID for VLAN networks or Tunnel ID for GENEVE/GRE/VXLAN networks

Returns
A generator of network objects

Return type
Network

update_network(*network*, *if_revision=None*, ***attrs*)

Update a network

Parameters

- **network** Either the id of a network or an instance of type *Network*.
- **if_revision** (*int*) Revision to put in If-Match header of update request to perform compare-and-swap update.
- **attrs** The attributes to update on the network represented by `network`.

Returns
The updated network

Return type
Network

find_network_ip_availability(*name_or_id*, *ignore_missing=True*, ***query*)

Find IP availability of a network

Parameters

- **name_or_id** The name or ID of a network.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns
One *NetworkIPAvailability* or None

get_network_ip_availability(*network*)

Get IP availability of a network

Parameters
network The value can be the ID of a network or a *Network* instance.

Returns
One *NetworkIPAvailability*

Raises
NotFoundException when no resource can be found.

network_ip_availabilities(***query*)

Return a generator of network ip availabilities

Parameters

query (*kwargs*) Optional query parameters to be sent to limit the resources being returned. Available parameters include:

- `ip_version`: IP version of the network
- `network_id`: ID of network to use when listening network IP availability.
- `network_name`: The name of the network for the particular network IP availability.
- `project_id`: Owner tenant ID

Returns

A generator of network ip availability objects

Return type

NetworkIPAvailability

Port Operations

class openstack.network.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_port(***attrs*)

Create a new port from attributes

Parameters

attrs Keyword arguments which will be used to create a *Port*, comprised of the properties on the Port class.

Returns

The results of port creation

Return type

Port

create_ports(*data*)

Create ports from the list of attributes

Parameters

data (*list*) List of dicts of attributes which will be used to create a *Port*, comprised of the properties on the Port class.

Returns

A generator of port objects

Return type

Port

delete_port(*port*, *ignore_missing=True*, *if_revision=None*)

Delete a port

Parameters

- **port** The value can be either the ID of a port or a *Port* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the port does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent port.
- **if_revision** (*int*) Revision to put in If-Match header of update request to perform compare-and-swap update.

Returns
`None`

find_port(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single port

Parameters

- **name_or_id** The name or ID of a port.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns
One *Port* or None

get_port(*port*)

Get a single port

Parameters
port The value can be the ID of a port or a *Port* instance.

Returns
One *Port*

Raises
NotFoundException when no resource can be found.

ports(***query*)

Return a generator of ports

Parameters
query (*kwargs*) Optional query parameters to be sent to limit the resources being returned. Available parameters include:

- `description`: The port description.
- `device_id`: Port device ID.
- `device_owner`: Port device owner (e.g. `network:dhcp`).
- `ip_address`: IP addresses of an allowed address pair.
- `is_admin_state_up`: The administrative state of the port.
- `is_port_security_enabled`: The port security status.

- `mac_address`: Port MAC address.
- `name`: The port name.
- `network_id`: ID of network that owns the ports.
- `project_id`: The ID of the project who owns the network.
- `status`: The port status. Value is `ACTIVE` or `DOWN`.
- `subnet_id`: The ID of the subnet.

Returns
A generator of port objects

Return type
Port

update_port(*port*, *if_revision=None*, ***attrs*)

Update a port

Parameters

- **port** Either the id of a port or a *Port* instance.
- **if_revision** (*int*) Revision to put in If-Match header of update request to perform compare-and-swap update.
- **attrs** The attributes to update on the port represented by `port`.

Returns
The updated port

Return type
Port

Router Operations

class openstack.network.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_router(***attrs*)

Create a new router from attributes

Parameters
attrs Keyword arguments which will be used to create a *Router*, comprised of the properties on the Router class.

Returns
The results of router creation

Return type
Router

delete_router(*router*, *ignore_missing=True*, *if_revision=None*)

Delete a router

Parameters

- **router** The value can be either the ID of a router or a *Router* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the router does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent router.
- **if_revision** (*int*) Revision to put in If-Match header of update request to perform compare-and-swap update.

Returns

None

find_router(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single router

Parameters

- **name_or_id** The name or ID of a router.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns

One *Router* or None

get_router(*router*)

Get a single router

Parameters

router The value can be the ID of a router or a *Router* instance.

Returns

One *Router*

Raises

NotFoundException when no resource can be found.

routers(***query*)

Return a generator of routers

Parameters

query (*dict*) Optional query parameters to be sent to limit the resources being returned. Valid parameters are:

- `description`: The description of a router.
- `flavor_id`: The ID of the flavor.
- `is_admin_state_up`: Router administrative state is up or not
- `is_distributed`: The distributed state of a router
- `is_ha`: The highly-available state of a router
- `name`: Router name

- `project_id`: The ID of the project this router is associated with.
- `status`: The status of the router.

Returns
A generator of router objects

Return type
Router

update_router(*router*, *if_revision=None*, ***attrs*)

Update a router

Parameters

- **router** Either the id of a router or a *Router* instance.
- **if_revision** (*int*) Revision to put in If-Match header of update request to perform compare-and-swap update.
- **attrs** The attributes to update on the router represented by `router`.

Returns
The updated router

Return type
Router

add_interface_to_router(*router*, *subnet_id=None*, *port_id=None*)

Add Interface to a router

Parameters

- **router** Either the router ID or an instance of *Router*
- **subnet_id** ID of the subnet
- **port_id** ID of the port

Returns
Router with updated interface

Return type
Router

remove_interface_from_router(*router*, *subnet_id=None*, *port_id=None*)

Remove Interface from a router

Parameters

- **router** Either the router ID or an instance of *Router*
- **subnet** ID of the subnet
- **port** ID of the port

Returns
Router with updated interface

Return type
Router

add_extra_routes_to_router(*router*, *body*)

Add extra routes to a router

Parameters

- **router** Either the router ID or an instance of *Router*
- **body** The request body as documented in the api-ref.

Returns

Router with updated extra routes

Return type

Router

remove_extra_routes_from_router(*router*, *body*)

Remove extra routes from a router

Parameters

- **router** Either the router ID or an instance of *Router*
- **body** The request body as documented in the api-ref.

Returns

Router with updated extra routes

Return type

Router

add_gateway_to_router(*router*, ***body*)

Add Gateway to a router

Parameters

- **router** Either the router ID or an instance of *Router*
- **body** Body with the gateway information

Returns

Router with updated interface

Return type

Router

remove_gateway_from_router(*router*, ***body*)

Remove Gateway from a router

Parameters

- **router** Either the router ID or an instance of *Router*
- **body** Body with the gateway information

Returns

Router with updated interface

Return type

Router

add_external_gateways(*router*, *body*)

Add router external gateways

Parameters

- **router** Either the router ID or an instance of *Router*
- **body** Body containing the external_gateways parameter.

Returns

Router with added gateways

Return type

Router

update_external_gateways(*router*, *body*)

Update router external gateways

Parameters

- **router** Either the router ID or an instance of *Router*
- **body** Body containing the external_gateways parameter.

Returns

Router with updated gateways

Return type

Router

remove_external_gateways(*router*, *body*)

Remove router external gateways

Parameters

- **router** Either the router ID or an instance of *Router*
- **body** Body containing the external_gateways parameter.

Returns

Router without the removed gateways

Return type

Router

create_conntrack_helper(*router*, ***attrs*)

Create a new L3 conntrack helper from attributes

Parameters

- **router** Either the router ID or an instance of *Router*
- **attrs** Keyword arguments which will be used to create a `ConntrackHelper`, comprised of the properties on the ConntrackHelper class.

Returns

The results of conntrack helper creation

Return type

`ConntrackHelper`

conntrack_helpers(*router*, ***query*)

Return a generator of conntrack helpers

Parameters

- **router** Either the router ID or an instance of *Router*
- **query** (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns

A generator of conntrack helper objects

Return type

ConntrackHelper

get_conntrack_helper(*conntrack_helper*, *router*)

Get a single L3 conntrack helper

Parameters

- **conntrack_helper** The value can be the ID of a L3 conntrack helper or a ConntrackHelper, instance.
- **router** The value can be the ID of a Router or a *Router* instance.

Returns

One ConntrackHelper

Raises

NotFoundException when no resource can be found.

update_conntrack_helper(*conntrack_helper*, *router*, ***attrs*)

Update a L3 conntrack_helper

Parameters

- **conntrack_helper** The value can be the ID of a L3 conntrack helper or a ConntrackHelper, instance.
- **router** The value can be the ID of a Router or a *Router* instance.
- **attrs** The attributes to update on the L3 conntrack helper represented by conntrack_helper.

Returns

The updated conntrack helper

Return type

ConntrackHelper

delete_conntrack_helper(*conntrack_helper*, *router*, *ignore_missing=True*)

Delete a L3 conntrack_helper

Parameters

- **conntrack_helper** The value can be the ID of a L3 conntrack helper or a ConntrackHelper, instance.
- **router** The value can be the ID of a Router or a *Router* instance.

- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the floating ip does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent ip.

Returns
`None`

Floating IP Operations

class openstack.network.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_ip(***attrs*)

Create a new floating ip from attributes

Parameters
attrs Keyword arguments which will be used to create a *FloatingIP*, comprised of the properties on the FloatingIP class.

Returns
The results of floating ip creation

Return type
FloatingIP

delete_ip(*floating_ip*, *ignore_missing=True*, *if_revision=None*)

Delete a floating ip

Parameters

- **floating_ip** The value can be either the ID of a floating ip or a *FloatingIP* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the floating ip does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent ip.
- **if_revision** (*int*) Revision to put in If-Match header of update request to perform compare-and-swap update.

Returns
`None`

find_available_ip()

Find an available IP

Returns
One *FloatingIP* or None

find_ip(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single IP

Parameters

- **name_or_id** The name or ID of an IP.

- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns
One *FloatingIP* or None

get_ip(*floating_ip*)

Get a single floating ip

Parameters
floating_ip The value can be the ID of a floating ip or a *FloatingIP* instance.

Returns
One *FloatingIP*

Raises
NotFoundException when no resource can be found.

ips(***query*)

Return a generator of ips

Parameters
query (*dict*) Optional query parameters to be sent to limit the resources being returned. Valid parameters are:

- `description`: The description of a floating IP.
- `fixed_ip_address`: The fixed IP address associated with a floating IP address.
- `floating_ip_address`: The IP address of a floating IP.
- `floating_network_id`: The ID of the network associated with a floating IP.
- `port_id`: The ID of the port to which a floating IP is associated.
- `project_id`: The ID of the project a floating IP is associated with.
- `router_id`: The ID of an associated router.
- `status`: The status of a floating IP, which can be `ACTIVE` or `DOWN`.

Returns
A generator of floating IP objects

Return type
FloatingIP

update_ip(*floating_ip*, *if_revision=None*, ***attrs*)

Update a ip

Parameters
- **floating_ip** Either the id of a ip or a *FloatingIP* instance.

- **if_revision** (*int*) Revision to put in If-Match header of update request to perform compare-and-swap update.
- **attrs** The attributes to update on the ip represented by `value`.

Returns
The updated ip

Return type
FloatingIP

Pool Operations

class openstack.network.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_pool(***attrs*)

Create a new pool from attributes

Parameters
attrs Keyword arguments which will be used to create a *Pool*, comprised of the properties on the Pool class.

Returns
The results of pool creation

Return type
Pool

delete_pool(*pool*, *ignore_missing=True*)

Delete a pool

Parameters

- **pool** The value can be either the ID of a pool or a *Pool* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the pool does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent pool.

Returns
`None`

find_pool(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single pool

Parameters

- **name_or_id** The name or ID of a pool.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns
One *Pool* or None

get_pool(*pool*)

Get a single pool

Parameters
pool The value can be the ID of a pool or a *Pool* instance.

Returns
One *Pool*

Raises
NotFoundException when no resource can be found.

pools(***query*)

Return a generator of pools

Parameters
query (*dict*) Optional query parameters to be sent to limit the resources being returned. Valid parameters are:

- `description`: The description for the pool.
- `is_admin_state_up`: The administrative state of the pool.
- `lb_algorithm`: The load-balancer algorithm used, which is one of `round-robin`, `least-connections` and so on.
- `name`: The name of the node pool.
- `project_id`: The ID of the project the pool is associated with.
- `protocol`: The protocol used by the pool, which is one of `TCP`, `HTTP` or `HTTPS`.
- `provider`: The name of the provider of the load balancer service.
- `subnet_id`: The subnet on which the members of the pool are located.
- `virtual_ip_id`: The ID of the virtual IP used.

Returns
A generator of pool objects

Return type
Pool

update_pool(*pool*, ***attrs*)

Update a pool

Parameters

- **pool** Either the id of a pool or a *Pool* instance.
- **attrs** The attributes to update on the pool represented by `pool`.

Returns
The updated pool

Return type
Pool

create_pool_member(*pool*, ***attrs*)

Create a new pool member from attributes

Parameters

- **pool** The pool can be either the ID of a pool or a *Pool* instance that the member will be created in.
- **attrs** Keyword arguments which will be used to create a *PoolMember*, comprised of the properties on the PoolMember class.

Returns

The results of pool member creation

Return type

PoolMember

delete_pool_member(*pool_member*, *pool*, *ignore_missing=True*)

Delete a pool member

Parameters

- **pool_member** The member can be either the ID of a pool member or a *PoolMember* instance.
- **pool** The pool can be either the ID of a pool or a *Pool* instance that the member belongs to.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the pool member does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent pool member.

Returns

`None`

find_pool_member(*name_or_id*, *pool*, *ignore_missing=True*, ***query*)

Find a single pool member

Parameters

- **name_or_id** (*str*) The name or ID of a pool member.
- **pool** The pool can be either the ID of a pool or a *Pool* instance that the member belongs to.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns

One *PoolMember* or None

get_pool_member(*pool_member*, *pool*)

Get a single pool member

Parameters

- **pool_member** The member can be the ID of a pool member or a *PoolMember* instance.
- **pool** The pool can be either the ID of a pool or a *Pool* instance that the member belongs to.

Returns
One *PoolMember*

Raises
NotFoundException when no resource can be found.

pool_members(*pool*, ***query*)

Return a generator of pool members

Parameters

- **pool** The pool can be either the ID of a pool or a *Pool* instance that the member belongs to.
- **query** (*dict*) Optional query parameters to be sent to limit the resources being returned. Valid parameters are:
  - `address`: The IP address of the pool member.
  - `is_admin_state_up`: The administrative state of the pool member.
  - `name`: Name of the pool member.
  - `project_id`: The ID of the project this pool member is associated with.
  - `protocol_port`: The port on which the application is hosted.
  - `subnet_id`: Subnet ID in which to access this pool member.
  - `weight`: A positive integer value that indicates the relative portion of traffic that this member should receive from the pool.

Returns
A generator of pool member objects

Return type
PoolMember

update_pool_member(*pool_member*, *pool*, ***attrs*)

Update a pool member

Parameters

- **pool_member** Either the ID of a pool member or a *PoolMember* instance.
- **pool** The pool can be either the ID of a pool or a *Pool* instance that the member belongs to.
- **attrs** The attributes to update on the pool member represented by `pool_member`.

Returns
The updated pool member

Return type
PoolMember

Auto Allocated Topology Operations

class openstack.network.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

get_auto_allocated_topology(*project=None*)

Get the auto-allocated topology of a given tenant

Parameters

project The value is the ID or name of a project

Returns

The auto-allocated topology

Return type

AutoAllocatedTopology

delete_auto_allocated_topology(*project=None*, *ignore_missing=False*)

Delete auto-allocated topology

Parameters

- **project** The value is the ID or name of a project
- **ignore_missing** When set to `False` *NotFoundException* will be raised when the topology does not exist. When set to `True`, no exception will be raised when attempting to delete nonexistant topology

Returns

None

validate_auto_allocated_topology(*project=None*)

Validate the resources for auto allocation

Parameters

project The value is the ID or name of a project

Returns

Whether all resources are correctly configured or not

Return type

ValidateTopology

Default Security Group Rules Operations

class openstack.network.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_default_security_group_rule(***attrs*)

Create a new default security group rule from attributes

Parameters
attrs Keyword arguments which will be used to create a `DefaultSecurityGroupRule`, comprised of the properties on the DefaultSecurityGroupRule class.

Returns
The results of default security group rule creation

Return type
`DefaultSecurityGroupRule`

delete_default_security_group_rule(*default_security_group_rule*, *ignore_missing=True*)

Delete a default security group rule

Parameters

- **default_security_group_rule** The value can be either the ID of a default security group rule or a `DefaultSecurityGroupRule` instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the defaul security group rule does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent default security group rule.

Returns
`None`

find_default_security_group_rule(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single default security group rule

Parameters

- **name_or_id** (*str*) The ID of a default security group rule.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns
One `DefaultSecurityGroupRule` or None

get_default_security_group_rule(*default_security_group_rule*)

Get a single default security group rule

Parameters
default_security_group_rule The value can be the ID of a default security group rule or a `DefaultSecurityGroupRule` instance.

Returns
`DefaultSecurityGroupRule`

Raises
NotFoundException when no resource can be found.

default_security_group_rules(***query*)

Return a generator of default security group rules

Parameters

query (*kwargs*) Optional query parameters to be sent to limit the resources being returned. Available parameters include:

- `description`: The default security group rule description
- `direction`: Default security group rule direction
- `ether_type`: Must be IPv4 or IPv6, and addresses represented in CIDR must match the ingress or egress rule.
- `protocol`: Default security group rule protocol
- `remote_group_id`: ID of a remote security group

Returns

A generator of default security group rule objects

Return type

`DefaultSecurityGroupRule`

Security Group Operations

class openstack.network.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_security_group(***attrs*)

Create a new security group from attributes

Parameters

attrs Keyword arguments which will be used to create a *SecurityGroup*, comprised of the properties on the SecurityGroup class.

Returns

The results of security group creation

Return type

SecurityGroup

delete_security_group(*security_group*, *ignore_missing=True*, *if_revision=None*)

Delete a security group

Parameters

- **security_group** The value can be either the ID of a security group or a *SecurityGroup* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the security group does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent security group.
- **if_revision** (*int*) Revision to put in If-Match header of update request to perform compare-and-swap update.

Returns
None

find_security_group(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single security group

Parameters

- **name_or_id** The name or ID of a security group.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns
One *SecurityGroup* or None

get_security_group(*security_group*)

Get a single security group

Parameters
security_group The value can be the ID of a security group or a *SecurityGroup* instance.

Returns
One *SecurityGroup*

Raises
NotFoundException when no resource can be found.

security_groups(***query*)

Return a generator of security groups

Parameters
query (*dict*) Optional query parameters to be sent to limit the resources being returned. Valid parameters are:

- `description`: Security group description
- `id`: The id of a security group, or list of security group ids
- `name`: The name of a security group
- `project_id`: The ID of the project this security group is associated with.

Returns
A generator of security group objects

Return type
SecurityGroup

update_security_group(*security_group*, *if_revision=None*, ***attrs*)

Update a security group

Parameters

- **security_group** Either the id of a security group or a *SecurityGroup* instance.

- **if_revision** (*int*) Revision to put in If-Match header of update request to perform compare-and-swap update.
- **attrs** The attributes to update on the security group represented by `security_group`.

Returns
The updated security group

Return type
SecurityGroup

create_security_group_rule(***attrs*)

Create a new security group rule from attributes

Parameters
attrs Keyword arguments which will be used to create a *SecurityGroupRule*, comprised of the properties on the SecurityGroupRule class.

Returns
The results of security group rule creation

Return type
SecurityGroupRule

create_security_group_rules(*data*)

Create new security group rules from the list of attributes

Parameters
data (*list*) List of dicts of attributes which will be used to create a *SecurityGroupRule*, comprised of the properties on the SecurityGroupRule class.

Returns
A generator of security group rule objects

Return type
SecurityGroupRule

delete_security_group_rule(*security_group_rule*, *ignore_missing=True*, *if_revision=None*)

Delete a security group rule

Parameters

- **security_group_rule** The value can be either the ID of a security group rule or a *SecurityGroupRule* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the security group rule does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent security group rule.
- **if_revision** (*int*) Revision to put in If-Match header of update request to perform compare-and-swap update.

Returns
`None`

find_security_group_rule(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single security group rule

Parameters

- **name_or_id** (*str*) The ID of a security group rule.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns

One *SecurityGroupRule* or None

get_security_group_rule(*security_group_rule*)

Get a single security group rule

Parameters

security_group_rule The value can be the ID of a security group rule or a *SecurityGroupRule* instance.

Returns

SecurityGroupRule

Raises

NotFoundException when no resource can be found.

security_group_rules(***query*)

Return a generator of security group rules

Parameters

query (*kwargs*) Optional query parameters to be sent to limit the resources being returned. Available parameters include:

- `description`: The security group rule description
- `direction`: Security group rule direction
- `ether_type`: Must be IPv4 or IPv6, and addresses represented in CIDR must match the ingress or egress rule.
- `project_id`: The ID of the project this security group rule is associated with.
- `protocol`: Security group rule protocol
- `remote_group_id`: ID of a remote security group
- `security_group_id`: ID of security group that owns the rules

Returns

A generator of security group rule objects

Return type

SecurityGroupRule

Address Group Operations

class openstack.network.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_address_group(***attrs*)

Create a new address group from attributes

Parameters

attrs Keyword arguments which will be used to create a *AddressGroup*, comprised of the properties on the AddressGroup class.

Returns

The results of address group creation

Return type

AddressGroup

delete_address_group(*address_group*, *ignore_missing=True*)

Delete an address group

Parameters

- **address_group** The value can be either the ID of an address group or a *AddressGroup* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the address group does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent address group.

Returns

`None`

find_address_group(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single address group

Parameters

- **name_or_id** The name or ID of an address group.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns

One *AddressGroup* or None

get_address_group(*address_group*)

Get a single address group

Parameters

address_group The value can be the ID of an address group or a *AddressGroup* instance.

Returns
One *AddressGroup*

Raises
NotFoundException when no resource can be found.

address_groups(***query*)

Return a generator of address groups

Parameters
query (*dict*) Optional query parameters to be sent to limit the resources being returned.

- `name`: Address group name
- `description`: Address group description
- `project_id`: Owner project ID

Returns
A generator of address group objects

Return type
AddressGroup

update_address_group(*address_group*, ***attrs*)

Update an address group

Parameters

- **address_group** Either the ID of an address group or a *AddressGroup* instance.
- **attrs** The attributes to update on the address group represented by `value`.

Returns
The updated address group

Return type
AddressGroup

add_addresses_to_address_group(*address_group*, *addresses*)

Add addresses to a address group

Parameters

- **address_group** Either the ID of an address group or a *AddressGroup* instance.
- **addresses** (*list*) List of address strings.

Returns
AddressGroup with updated addresses

Return type
AddressGroup

remove_addresses_from_address_group(*address_group*, *addresses*)

Remove addresses from a address group

Parameters

- **address_group** Either the ID of an address group or a *AddressGroup* instance.
- **addresses** (*list*) List of address strings.

Returns
AddressGroup with updated addresses

Return type
AddressGroup

Availability Zone Operations

class openstack.network.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

availability_zones(***query*)

Return a generator of availability zones

Parameters
query (*dict*) optional query parameters to be set to limit the returned resources. Valid parameters include:

- name: The name of an availability zone.
- resource: The type of resource for the availability zone.

Returns
A generator of availability zone objects

Return type
AvailabilityZone

Address Scope Operations

class openstack.network.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_address_scope(***attrs*)

Create a new address scope from attributes

Parameters
attrs Keyword arguments which will be used to create a *AddressScope*, comprised of the properties on the AddressScope class.

Returns
The results of address scope creation

Return type
AddressScope

delete_address_scope(*address_scope*, *ignore_missing=True*)

Delete an address scope

Parameters

- **address_scope** The value can be either the ID of an address scope or a *AddressScope* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the address scope does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent address scope.

Returns

`None`

find_address_scope(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single address scope

Parameters

- **name_or_id** The name or ID of an address scope.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns

One *AddressScope* or None

get_address_scope(*address_scope*)

Get a single address scope

Parameters

address_scope The value can be the ID of an address scope or a *AddressScope* instance.

Returns

One *AddressScope*

Raises

NotFoundException when no resource can be found.

address_scopes(***query*)

Return a generator of address scopes

Parameters

query (*dict*) Optional query parameters to be sent to limit the resources being returned.

- `name`: Address scope name
- `ip_version`: Address scope IP address version
- `tenant_id`: Owner tenant ID
- `shared`: Address scope is shared (boolean)

Returns
A generator of address scope objects

Return type
AddressScope

update_address_scope(*address_scope*, ***attrs*)
Update an address scope

Parameters

- **address_scope** Either the ID of an address scope or a *AddressScope* instance.
- **attrs** The attributes to update on the address scope represented by `value`.

Returns
The updated address scope

Return type
AddressScope

Quota Operations

class openstack.network.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

delete_quota(*quota*, *ignore_missing=True*)
Delete a quota (i.e. reset to the default quota)

Parameters

- **quota** The value can be either the ID of a quota or a *Quota* instance. The ID of a quota is the same as the project ID for the quota.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when quota does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent quota.

Returns
`None`

get_quota(*quota*, *details=False*)
Get a quota

Parameters

- **quota** The value can be the ID of a quota or a *Quota* instance. The ID of a quota is the same as the project ID for the quota.
- **details** If set to True, details about quota usage will be returned.

Returns
One *Quota*

Raises
NotFoundException when no resource can be found.

get_quota_default(*quota*)

Get a default quota

Parameters

quota The value can be the ID of a default quota or a `QuotaDefault` instance. The ID of a default quota is the same as the project ID for the default quota.

Returns

One `QuotaDefault`

Raises

NotFoundException when no resource can be found.

quotas(***query*)

Return a generator of quotas

Parameters

query (*dict*) Optional query parameters to be sent to limit the resources being returned. Currently no query parameter is supported.

Returns

A generator of quota objects

Return type

Quota

update_quota(*quota*, ***attrs*)

Update a quota

Parameters

- **quota** Either the ID of a quota or a *Quota* instance. The ID of a quota is the same as the project ID for the quota.
- **attrs** The attributes to update on the quota represented by `quota`.

Returns

The updated quota

Return type

Quota

QoS Operations

class openstack.network.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_qos_bandwidth_limit_rule(*qos_policy*, ***attrs*)

Create a new bandwidth limit rule

Parameters

- **attrs** Keyword arguments which will be used to create a *QoSBandwidthLimitRule*, comprised of the properties on the QoS-BandwidthLimitRule class.

- **qos_policy** The value can be the ID of the QoS policy that the rule belongs or a *QoSPolicy* instance.

Returns
The results of resource creation

Return type
QoSBandwidthLimitRule

delete_qos_bandwidth_limit_rule(*qos_rule*, *qos_policy*, *ignore_missing=True*)

Delete a bandwidth limit rule

Parameters

- **qos_rule** The value can be either the ID of a bandwidth limit rule or a *QoSBandwidthLimitRule* instance.
- **qos_policy** The value can be the ID of the QoS policy that the rule belongs or a *QoSPolicy* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent bandwidth limit rule.

Returns
`None`

find_qos_bandwidth_limit_rule(*qos_rule_id*, *qos_policy*, *ignore_missing=True*, ***query*)

Find a bandwidth limit rule

Parameters

- **qos_rule_id** The ID of a bandwidth limit rule.
- **qos_policy** The value can be the ID of the QoS policy that the rule belongs or a *QoSPolicy* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns
One *QoSBandwidthLimitRule* or None

get_qos_bandwidth_limit_rule(*qos_rule*, *qos_policy*)

Get a single bandwidth limit rule

Parameters

- **qos_rule** The value can be the ID of a minimum bandwidth rule or a *QoSBandwidthLimitRule* instance.
- **qos_policy** The value can be the ID of the QoS policy that the rule belongs or a *QoSPolicy* instance.

Returns
One *QoSBandwidthLimitRule*

Raises
NotFoundException when no resource can be found.

qos_bandwidth_limit_rules(*qos_policy*, ***query*)

Return a generator of bandwidth limit rules

Parameters

- **qos_policy** The value can be the ID of the QoS policy that the rule belongs or a *QoSPolicy* instance.
- **query** (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of bandwidth limit rule objects

Return type
QoSBandwidthLimitRule

update_qos_bandwidth_limit_rule(*qos_rule*, *qos_policy*, ***attrs*)

Update a bandwidth limit rule

Parameters

- **qos_rule** Either the id of a bandwidth limit rule or a *QoSBandwidthLimitRule* instance.
- **qos_policy** The value can be the ID of the QoS policy that the rule belongs or a *QoSPolicy* instance.
- **attrs** The attributes to update on the bandwidth limit rule represented by `qos_rule`.

Returns
The updated minimum bandwidth rule

Return type
QoSBandwidthLimitRule

create_qos_dscp_marking_rule(*qos_policy*, ***attrs*)

Create a new QoS DSCP marking rule

Parameters

- **attrs** Keyword arguments which will be used to create a *QoSDSCPMarkingRule*, comprised of the properties on the QosDscpMarkingRule class.
- **qos_policy** The value can be the ID of the QoS policy that the rule belongs or a *QoSPolicy* instance.

Returns
The results of router creation

Return type
QoSDSCPMarkingRule

delete_qos_dscp_marking_rule(*qos_rule*, *qos_policy*, *ignore_missing=True*)

Delete a QoS DSCP marking rule

Parameters

- **qos_rule** The value can be either the ID of a minimum bandwidth rule or a *QoSDSCPMarkingRule* instance.
- **qos_policy** The value can be the ID of the QoS policy that the rule belongs or a *QoSPolicy* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent minimum bandwidth rule.

Returns

`None`

find_qos_dscp_marking_rule(*qos_rule_id*, *qos_policy*, *ignore_missing=True*, ***query*)

Find a QoS DSCP marking rule

Parameters

- **qos_rule_id** The ID of a QoS DSCP marking rule.
- **qos_policy** The value can be the ID of the QoS policy that the rule belongs or a *QoSPolicy* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns

One *QoSDSCPMarkingRule* or None

get_qos_dscp_marking_rule(*qos_rule*, *qos_policy*)

Get a single QoS DSCP marking rule

Parameters

- **qos_rule** The value can be the ID of a minimum bandwidth rule or a *QoSDSCPMarkingRule* instance.
- **qos_policy** The value can be the ID of the QoS policy that the rule belongs or a *QoSPolicy* instance.

Returns

One *QoSDSCPMarkingRule*

Raises

NotFoundException when no resource can be found.

qos_dscp_marking_rules(*qos_policy*, ***query*)

Return a generator of QoS DSCP marking rules

Parameters

- **qos_policy** The value can be the ID of the QoS policy that the rule belongs or a *QoSPolicy* instance.

- **query** (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of QoS DSCP marking rule objects

Return type
QoSDSCPMarkingRule

update_qos_dscp_marking_rule(*qos_rule*, *qos_policy*, ***attrs*)

Update a QoS DSCP marking rule

Parameters

- **qos_rule** Either the id of a minimum bandwidth rule or a *QoSDSCPMarkingRule* instance.
- **qos_policy** The value can be the ID of the QoS policy that the rule belongs or a *QoSPolicy* instance.
- **attrs** The attributes to update on the QoS DSCP marking rule represented by `qos_rule`.

Returns
The updated QoS DSCP marking rule

Return type
QoSDSCPMarkingRule

create_qos_minimum_bandwidth_rule(*qos_policy*, ***attrs*)

Create a new minimum bandwidth rule

Parameters

- **attrs** Keyword arguments which will be used to create a *QoSMinimumBandwidthRule*, comprised of the properties on the QoSMinimumBandwidthRule class.
- **qos_policy** The value can be the ID of the QoS policy that the rule belongs or a *QoSPolicy* instance.

Returns
The results of resource creation

Return type
QoSMinimumBandwidthRule

delete_qos_minimum_bandwidth_rule(*qos_rule*, *qos_policy*, *ignore_missing=True*)

Delete a minimum bandwidth rule

Parameters

- **qos_rule** The value can be either the ID of a minimum bandwidth rule or a *QoSMinimumBandwidthRule* instance.
- **qos_policy** The value can be the ID of the QoS policy that the rule belongs or a *QoSPolicy* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent minimum bandwidth rule.

Returns
None

find_qos_minimum_bandwidth_rule(*qos_rule_id*, *qos_policy*, *ignore_missing=True*, ***query*)

Find a minimum bandwidth rule

Parameters

- **qos_rule_id** The ID of a minimum bandwidth rule.
- **qos_policy** The value can be the ID of the QoS policy that the rule belongs or a *QoSPolicy* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns
One *QoSMinimumBandwidthRule* or None

get_qos_minimum_bandwidth_rule(*qos_rule*, *qos_policy*)

Get a single minimum bandwidth rule

Parameters

- **qos_rule** The value can be the ID of a minimum bandwidth rule or a *QoSMinimumBandwidthRule* instance.
- **qos_policy** The value can be the ID of the QoS policy that the rule belongs or a *QoSPolicy* instance.

Returns
One *QoSMinimumBandwidthRule*

Raises
NotFoundException when no resource can be found.

qos_minimum_bandwidth_rules(*qos_policy*, ***query*)

Return a generator of minimum bandwidth rules

Parameters

- **qos_policy** The value can be the ID of the QoS policy that the rule belongs or a *QoSPolicy* instance.
- **query** (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of minimum bandwidth rule objects

Return type
QoSMinimumBandwidthRule

update_qos_minimum_bandwidth_rule(*qos_rule*, *qos_policy*, ***attrs*)

Update a minimum bandwidth rule

Parameters

- **qos_rule** Either the id of a minimum bandwidth rule or a *QoSMinimumBandwidthRule* instance.
- **qos_policy** The value can be the ID of the QoS policy that the rule belongs or a *QoSPolicy* instance.
- **attrs** The attributes to update on the minimum bandwidth rule represented by qos_rule.

Returns
The updated minimum bandwidth rule

Return type
QoSMinimumBandwidthRule

create_qos_minimum_packet_rate_rule(*qos_policy*, ***attrs*)

Create a new minimum packet rate rule

Parameters

- **attrs** Keyword arguments which will be used to create a *QoSMinimumPacketRateRule*, comprised of the properties on the QoSMinimumPacketRateRule class.
- **qos_policy** The value can be the ID of the QoS policy that the rule belongs or a *QoSPolicy* instance.

Returns
The results of resource creation

Return type
QoSMinimumPacketRateRule

delete_qos_minimum_packet_rate_rule(*qos_rule*, *qos_policy*, *ignore_missing=True*)

Delete a minimum packet rate rule

Parameters

- **qos_rule** The value can be either the ID of a minimum packet rate rule or a *QoSMinimumPacketRateRule* instance.
- **qos_policy** The value can be the ID of the QoS policy that the rule belongs or a *QoSPolicy* instance.
- **ignore_missing** (*bool*) When set to False *NotFoundException* will be raised when the resource does not exist. When set to True, no exception will be set when attempting to delete a nonexistent minimum packet rate rule.

Returns
None

find_qos_minimum_packet_rate_rule(*qos_rule_id*, *qos_policy*, *ignore_missing=True*, ***query*)

Find a minimum packet rate rule

Parameters

- **qos_rule_id** The ID of a minimum packet rate rule.

- **qos_policy** The value can be the ID of the QoS policy that the rule belongs or a *QoSPolicy* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns
One *QoSMinimumPacketRateRule* or None

get_qos_minimum_packet_rate_rule(*qos_rule*, *qos_policy*)

Get a single minimum packet rate rule

Parameters

- **qos_rule** The value can be the ID of a minimum packet rate rule or a *QoSMinimumPacketRateRule* instance.
- **qos_policy** The value can be the ID of the QoS policy that the rule belongs or a *QoSPolicy* instance.

Returns
One *QoSMinimumPacketRateRule*

Raises
NotFoundException when no resource can be found.

qos_minimum_packet_rate_rules(*qos_policy*, ***query*)

Return a generator of minimum packet rate rules

Parameters

- **qos_policy** The value can be the ID of the QoS policy that the rule belongs or a *QoSPolicy* instance.
- **query** (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of minimum packet rate rule objects

Return type
QoSMinimumPacketRateRule

update_qos_minimum_packet_rate_rule(*qos_rule*, *qos_policy*, ***attrs*)

Update a minimum packet rate rule

Parameters

- **qos_rule** Either the id of a minimum packet rate rule or a *QoSMinimumPacketRateRule* instance.
- **qos_policy** The value can be the ID of the QoS policy that the rule belongs or a *QoSPolicy* instance.
- **attrs** The attributes to update on the minimum packet rate rule represented by `qos_rule`.

Returns
The updated minimum packet rate rule

Return type
QoSMinimumPacketRateRule

create_qos_policy(***attrs*)

Create a new QoS policy from attributes

Parameters
attrs Keyword arguments which will be used to create a *QoSPolicy*, comprised of the properties on the QoSPolicy class.

Returns
The results of QoS policy creation

Return type
QoSPolicy

delete_qos_policy(*qos_policy*, *ignore_missing=True*)

Delete a QoS policy

Parameters

- **qos_policy** The value can be either the ID of a QoS policy or a *QoSPolicy* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the QoS policy does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent QoS policy.

Returns
None

find_qos_policy(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single QoS policy

Parameters

- **name_or_id** The name or ID of a QoS policy.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns
One *QoSPolicy* or None

get_qos_policy(*qos_policy*)

Get a single QoS policy

Parameters
qos_policy The value can be the ID of a QoS policy or a *QoSPolicy* instance.

Returns
One *QoSPolicy*

Raises
NotFoundException when no resource can be found.

qos_policies(***query*)

Return a generator of QoS policies

Parameters
query (*dict*) Optional query parameters to be sent to limit the resources being returned. Valid parameters are:

- `description`: The description of a QoS policy.
- `is_shared`: Whether the policy is shared among projects.
- `name`: The name of a QoS policy.
- `project_id`: The ID of the project who owns the network.

Returns
A generator of QoS policy objects

Return type
QoSPolicy

update_qos_policy(*qos_policy*, ***attrs*)

Update a QoS policy

Parameters

- **qos_policy** Either the id of a QoS policy or a *QoSPolicy* instance.
- **attrs** The attributes to update on the QoS policy represented by `qos_policy`.

Returns
The updated QoS policy

Return type
QoSPolicy

find_qos_rule_type(*rule_type_name*, *ignore_missing=True*)

Find a single QoS rule type details

Parameters

- **rule_type_name** The name of a QoS rule type.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns
One *QoSRuleType* or None

get_qos_rule_type(*qos_rule_type*)

Get details about single QoS rule type

Parameters
qos_rule_type The value can be the name of a QoS policy rule type or a *QoSRuleType* instance.

Returns
One *QoSRuleType*

Raises
NotFoundException when no resource can be found.

`qos_rule_types`(***query*)

Return a generator of QoS rule types

Parameters
query (*dict*) Optional query parameters to be sent to limit the resources returned. Valid parameters include:

- `type`: The type of the QoS rule type.

Returns
A generator of QoS rule type objects

Return type
QoSRuleType

Agent Operations

`class openstack.network.v2._proxy.Proxy`(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

`agents`(***query*)

Return a generator of network agents

Parameters
query (*dict*) Optional query parameters to be sent to limit the resources being returned.

- `agent_type`: Agent type.
- `availability_zone`: The availability zone for an agent.
- `binary`: The name of the agents application binary.
- `description`: The description of the agent.
- `host`: The host (host name or host address) the agent is running on.
- `topic`: The message queue topic used.
- `is_admin_state_up`: The administrative state of the agent.
- `is_alive`: Whether the agent is alive.

Returns
A generator of agents

Return type
Agent

delete_agent(*agent*, *ignore_missing=True*)

Delete a network agent

Parameters

- **agent** The value can be the ID of a agent or a *Agent* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the agent does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent agent.

Returns

`None`

get_agent(*agent*)

Get a single network agent

Parameters

agent The value can be the ID of a agent or a *Agent* instance.

Returns

One *Agent*

Return type

Agent

Raises

NotFoundException when no resource can be found.

update_agent(*agent*, ***attrs*)

Update a network agent

Parameters

- **agent** The value can be the ID of a agent or a *Agent* instance.
- **attrs** The attributes to update on the agent represented by `value`.

Returns

One *Agent*

Return type

Agent

network_hosting_dhcp_agents(*network*, ***query*)

A generator of DHCP agents hosted on a network.

Parameters

- **network** The instance of *Network*
- **query** (*dict*) Optional query parameters to be sent to limit the resources returned.

Returns

A generator of hosted DHCP agents

routers_hosting_l3_agents(*router*, ***query*)

Return a generator of L3 agent hosting a router

Parameters

- **router** Either the router id or an instance of *Router*
- **query** (*kwargs*) Optional query parameters to be sent to limit the resources returned

Returns
A generator of Router L3 Agents

Return type
RouterL3Agents

agent_hosted_routers(*agent*, ***query*)

Return a generator of routers hosted by a L3 agent

Parameters

- **agent** Either the agent id of an instance of Agent
- **query** (*kwargs*) Optional query parameters to be sent to limit the resources returned

Returns
A generator of routers

Return type
L3AgentRouters

add_router_to_agent(*agent*, *router*)

Add router to L3 agent

Parameters

- **agent** Either the id of an agent *Agent* instance
- **router** A router instance

Returns
Agent with attached router

Return type
Agent

remove_router_from_agent(*agent*, *router*)

Remove router from L3 agent

Parameters

- **agent** Either the id of an agent or an *Agent* instance
- **router** A router instance

Returns
Agent with removed router

Return type
Agent

RBAC Operations

class openstack.network.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_rbac_policy(***attrs*)

Create a new RBAC policy from attributes

Parameters

attrs Keyword arguments which will be used to create a *RBACPolicy*, comprised of the properties on the RBACPolicy class.

Returns

The results of RBAC policy creation

Return type

RBACPolicy

delete_rbac_policy(*rbac_policy*, *ignore_missing=True*)

Delete a RBAC policy

Parameters

- **rbac_policy** The value can be either the ID of a RBAC policy or a *RBACPolicy* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the RBAC policy does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent RBAC policy.

Returns

`None`

find_rbac_policy(*rbac_policy*, *ignore_missing=True*, ***query*)

Find a single RBAC policy

Parameters

- **rbac_policy** The ID of a RBAC policy.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns

One *RBACPolicy* or None

get_rbac_policy(*rbac_policy*)

Get a single RBAC policy

Parameters

rbac_policy The value can be the ID of a RBAC policy or a *RBACPolicy* instance.

Returns
One *RBACPolicy*

Raises
NotFoundException when no resource can be found.

rbac_policies(***query*)

Return a generator of RBAC policies

Parameters
query (*dict*) Optional query parameters to be sent to limit the resources being returned. Available parameters include:

- `action`: RBAC policy action
- `object_type`: Type of the object that the RBAC policy affects
- `target_project_id`: ID of the tenant that the RBAC policy affects
- `project_id`: Owner tenant ID

Returns
A generator of rbac objects

Return type
RBACPolicy

update_rbac_policy(*rbac_policy*, ***attrs*)

Update a RBAC policy

Parameters
- **rbac_policy** Either the id of a RBAC policy or a *RBACPolicy* instance.
- **attrs** The attributes to update on the RBAC policy represented by `rbac_policy`.

Returns
The updated RBAC policy

Return type
RBACPolicy

Listener Operations

class openstack.network.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_listener(***attrs*)

Create a new listener from attributes

Parameters
attrs Keyword arguments which will be used to create a *Listener*, comprised of the properties on the Listener class.

Returns
The results of listener creation

Return type
Listener

delete_listener(*listener*, *ignore_missing=True*)

Delete a listener

Parameters

- **listener** The value can be either the ID of a listner or a *Listener* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the listner does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent listener.

Returns
`None`

find_listener(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single listener

Parameters

- **name_or_id** The name or ID of a listener.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns
One *Listener* or None

get_listener(*listener*)

Get a single listener

Parameters
listener The value can be the ID of a listener or a *Listener* instance.

Returns
One *Listener*

Raises
NotFoundException when no resource can be found.

listeners(***query*)

Return a generator of listeners

Parameters
query (*dict*) Optional query parameters to be sent to limit the resources being returned. Valid parameters are:

- `connection_limit`: The maximum number of connections permitted for the load-balancer.
- `default_pool_id`: The ID of the default pool.
- `default_tls_container_ref`: A reference to a container of TLS secret.
- `description`: The description of a listener.

- `is_admin_state_up`: The administrative state of the listener.
- `name`: The name of a listener.
- `project_id`: The ID of the project associated with a listener.
- `protocol`: The protocol of the listener.
- `protocol_port`: Port the listener will listen to.

Returns
A generator of listener objects

Return type
Listener

update_listener(*listener*, ***attrs*)

Update a listener

Parameters

- **listener** Either the id of a listener or a *Listener* instance.
- **attrs** The attributes to update on the listener represented by `listener`.

Returns
The updated listener

Return type
Listener

Subnet Operations

class openstack.network.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_subnet(***attrs*)

Create a new subnet from attributes

Parameters
attrs Keyword arguments which will be used to create a *Subnet*, comprised of the properties on the Subnet class.

Returns
The results of subnet creation

Return type
Subnet

delete_subnet(*subnet*, *ignore_missing=True*, *if_revision=None*)

Delete a subnet

Parameters

- **subnet** The value can be either the ID of a subnet or a *Subnet* instance.

- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the subnet does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent subnet.
- **if_revision** (*int*) Revision to put in If-Match header of update request to perform compare-and-swap update.

Returns
`None`

find_subnet(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single subnet

Parameters

- **name_or_id** The name or ID of a subnet.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns
One *Subnet* or None

get_subnet(*subnet*)

Get a single subnet

Parameters
subnet The value can be the ID of a subnet or a *Subnet* instance.

Returns
One *Subnet*

Raises
NotFoundException when no resource can be found.

subnets(***query*)

Return a generator of subnets

Parameters
query (*dict*) Optional query parameters to be sent to limit the resources being returned. Available parameters include:

- `cidr`: Subnet CIDR
- `description`: The subnet description
- `gateway_ip`: Subnet gateway IP address
- `ip_version`: Subnet IP address version
- `ipv6_address_mode`: The IPv6 address mode
- `ipv6_ra_mode`: The IPv6 router advertisement mode
- `is_dhcp_enabled`: Subnet has DHCP enabled (boolean)
- `name`: Subnet name

- `network_id`: ID of network that owns the subnets
- `project_id`: Owner tenant ID
- `subnet_pool_id`: The subnet pool ID from which to obtain a CIDR.

Returns
A generator of subnet objects

Return type
Subnet

update_subnet(*subnet*, *if_revision=None*, ***attrs*)

Update a subnet

Parameters

- **subnet** Either the id of a subnet or a *Subnet* instance.
- **if_revision** (*int*) Revision to put in If-Match header of update request to perform compare-and-swap update.
- **attrs** The attributes to update on the subnet represented by `subnet`.

Returns
The updated subnet

Return type
Subnet

create_subnet_pool(***attrs*)

Create a new subnet pool from attributes

Parameters
attrs Keyword arguments which will be used to create a *SubnetPool*, comprised of the properties on the SubnetPool class.

Returns
The results of subnet pool creation

Return type
SubnetPool

delete_subnet_pool(*subnet_pool*, *ignore_missing=True*)

Delete a subnet pool

Parameters

- **subnet_pool** The value can be either the ID of a subnet pool or a *SubnetPool* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the subnet pool does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent subnet pool.

Returns
`None`

find_subnet_pool(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single subnet pool

Parameters

- **name_or_id** The name or ID of a subnet pool.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns
One *SubnetPool* or None

get_subnet_pool(*subnet_pool*)

Get a single subnet pool

Parameters
subnet_pool The value can be the ID of a subnet pool or a *SubnetPool* instance.

Returns
One *SubnetPool*

Raises
NotFoundException when no resource can be found.

subnet_pools(***query*)

Return a generator of subnet pools

Parameters
query (*kwargs*) Optional query parameters to be sent to limit the resources being returned. Available parameters include:

- `address_scope_id`: Subnet pool address scope ID
- `description`: The subnet pool description
- `ip_version`: The IP address family
- `is_default`: Subnet pool is the default (boolean)
- `is_shared`: Subnet pool is shared (boolean)
- `name`: Subnet pool name
- `project_id`: Owner tenant ID

Returns
A generator of subnet pool objects

Return type
SubnetPool

update_subnet_pool(*subnet_pool*, ***attrs*)

Update a subnet pool

Parameters

- **subnet_pool** Either the ID of a subnet pool or a *SubnetPool* instance.

- **attrs** The attributes to update on the subnet pool represented by subnet_pool.

Returns
The updated subnet pool

Return type
SubnetPool

Load Balancer Operations

class openstack.network.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_load_balancer(***attrs*)

Create a new load balancer from attributes

Parameters
attrs Keyword arguments which will be used to create a *LoadBalancer*, comprised of the properties on the LoadBalancer class.

Returns
The results of load balancer creation

Return type
LoadBalancer

delete_load_balancer(*load_balancer*, *ignore_missing=True*)

Delete a load balancer

Parameters

- **load_balancer** The value can be the ID of a load balancer or a *LoadBalancer* instance.
- **ignore_missing** (*bool*) When set to False *NotFoundException* will be raised when the load balancer does not exist. When set to True, no exception will be set when attempting to delete a nonexistent load balancer.

Returns
None

find_load_balancer(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single load balancer

Parameters

- **name_or_id** The name or ID of a load balancer.
- **ignore_missing** (*bool*) When set to False *NotFoundException* will be raised when the resource does not exist. When set to True, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns
One *LoadBalancer* or None

get_load_balancer(*load_balancer*)

Get a single load balancer

Parameters
load_balancer The value can be the ID of a load balancer or a *LoadBalancer* instance.

Returns
One *LoadBalancer*

Raises
NotFoundException when no resource can be found.

load_balancers(***query*)

Return a generator of load balancers

Parameters
query (*dict*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of load balancer objects

Return type
LoadBalancer

update_load_balancer(*load_balancer*, ***attrs*)

Update a load balancer

Parameters

- **load_balancer** Either the id of a load balancer or a *LoadBalancer* instance.
- **attrs** The attributes to update on the load balancer represented by `load_balancer`.

Returns
The updated load balancer

Return type
LoadBalancer

Health Monitor Operations

class openstack.network.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_health_monitor(***attrs*)

Create a new health monitor from attributes

Parameters
attrs Keyword arguments which will be used to create a *HealthMonitor*, comprised of the properties on the HealthMonitor class.

Returns
The results of health monitor creation

Return type
HealthMonitor

delete_health_monitor(*health_monitor*, *ignore_missing=True*)

Delete a health monitor

Parameters

- **health_monitor** The value can be either the ID of a health monitor or a *HealthMonitor* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the health monitor does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent health monitor.

Returns
`None`

find_health_monitor(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single health monitor

Parameters

- **name_or_id** The name or ID of a health monitor.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns
One *HealthMonitor* or None

get_health_monitor(*health_monitor*)

Get a single health monitor

Parameters
health_monitor The value can be the ID of a health monitor or a *HealthMonitor* instance.

Returns
One *HealthMonitor*

Raises
NotFoundException when no resource can be found.

health_monitors(***query*)

Return a generator of health monitors

Parameters

query (*dict*) Optional query parameters to be sent to limit the resources being returned. Valid parameters are:

- `delay`: the time in milliseconds between sending probes.
- `expected_codes`: The expected HTTP codes for a pssing HTTP(S) monitor.
- `http_method`: The HTTP method a monitor uses for requests.
- `is_admin_state_up`: The administrative state of a health monitor.
- `max_retries`: The maximum consecutive health probe attempts.
- `project_id`: The ID of the project this health monitor is associated with.
- `timeout`: The maximum number of milliseconds for a monitor to wait for a connection to be established before it times out.
- `type`: The type of probe sent by the load balancer for health check, which can be `PING`, `TCP`, `HTTP` or `HTTPS`.
- `url_path`: The path portion of a URI that will be probed.

Returns

A generator of health monitor objects

Return type

HealthMonitor

update_health_monitor(*health_monitor*, ***attrs*)

Update a health monitor

Parameters

- **health_monitor** Either the id of a health monitor or a *HealthMonitor* instance.
- **attrs** The attributes to update on the health monitor represented by `value`.

Returns

The updated health monitor

Return type

HealthMonitor

Metering Label Operations

class openstack.network.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_metering_label(***attrs*)

Create a new metering label from attributes

Parameters
attrs Keyword arguments which will be used to create a *MeteringLabel*, comprised of the properties on the MeteringLabel class.

Returns
The results of metering label creation

Return type
MeteringLabel

delete_metering_label(*metering_label*, *ignore_missing=True*)

Delete a metering label

Parameters

- **metering_label** The value can be either the ID of a metering label or a *MeteringLabel* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the metering label does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent metering label.

Returns
`None`

find_metering_label(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single metering label

Parameters

- **name_or_id** The name or ID of a metering label.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns
One *MeteringLabel* or None

get_metering_label(*metering_label*)

Get a single metering label

Parameters
metering_label The value can be the ID of a metering label or a *MeteringLabel* instance.

Returns
One *MeteringLabel*

Raises
NotFoundException when no resource can be found.

metering_labels(***query*)

Return a generator of metering labels

Parameters

query (*dict*) Optional query parameters to be sent to limit the resources being returned. Valid parameters are:

- `description`: Description of a metering label.
- `name`: Name of a metering label.
- `is_shared`: Boolean indicating whether a metering label is shared.
- `project_id`: The ID of the project a metering label is associated with.

Returns

A generator of metering label objects

Return type

MeteringLabel

update_metering_label(*metering_label*, ***attrs*)

Update a metering label

Parameters

- **metering_label** Either the id of a metering label or a *MeteringLabel* instance.
- **attrs** The attributes to update on the metering label represented by `metering_label`.

Returns

The updated metering label

Return type

MeteringLabel

create_metering_label_rule(***attrs*)

Create a new metering label rule from attributes

Parameters

attrs Keyword arguments which will be used to create a *MeteringLabelRule*, comprised of the properties on the MeteringLabelRule class.

Returns

The results of metering label rule creation

Return type

MeteringLabelRule

delete_metering_label_rule(*metering_label_rule*, *ignore_missing=True*)

Delete a metering label rule

Parameters

- **metering_label_rule** The value can be either the ID of a metering label rule or a *MeteringLabelRule* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the metering label rule does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent metering label rule.

Returns
None

`find_metering_label_rule`(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single metering label rule

Parameters

- **name_or_id** The name or ID of a metering label rule.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns
One *MeteringLabelRule* or None

`get_metering_label_rule`(*metering_label_rule*)

Get a single metering label rule

Parameters
metering_label_rule The value can be the ID of a metering label rule or a *MeteringLabelRule* instance.

Returns
One *MeteringLabelRule*

Raises
NotFoundException when no resource can be found.

`metering_label_rules`(***query*)

Return a generator of metering label rules

Parameters
query (*dict*) Optional query parameters to be sent to limit the resources being returned. Valid parameters are:

- `direction`: The direction in which metering label rule is applied.
- `metering_label_id`: The ID of a metering label this rule is associated with.
- `project_id`: The ID of the project the metering label rule is associated with.
- `remote_ip_prefix`: The remote IP prefix to be associated with this metering label rule.

Returns
A generator of metering label rule objects

Return type
MeteringLabelRule

`update_metering_label_rule`(*metering_label_rule*, ***attrs*)

Update a metering label rule

Parameters

- **metering_label_rule** Either the id of a metering label rule or a *MeteringLabelRule* instance.
- **attrs** The attributes to update on the metering label rule represented by `metering_label_rule`.

Returns
The updated metering label rule

Return type
MeteringLabelRule

Segment Operations

class openstack.network.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_segment(***attrs*)

Create a new segment from attributes

Parameters
attrs Keyword arguments which will be used to create a *Segment*, comprised of the properties on the Segment class.

Returns
The results of segment creation

Return type
Segment

delete_segment(*segment*, *ignore_missing=True*)

Delete a segment

Parameters

- **segment** The value can be either the ID of a segment or a *Segment* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the segment does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent segment.

Returns
`None`

find_segment(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single segment

Parameters

- **name_or_id** The name or ID of a segment.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns
One *Segment* or None

`get_segment`(*segment*)

Get a single segment

Parameters
segment The value can be the ID of a segment or a *Segment* instance.

Returns
One *Segment*

Raises
NotFoundException when no resource can be found.

`segments`(***query*)

Return a generator of segments

Parameters
query (*kwargs*) Optional query parameters to be sent to limit the resources being returned. Available parameters include:

- `description`: The segment description
- `name`: Name of the segments
- `network_id`: ID of the network that owns the segments
- `network_type`: Network type for the segments
- `physical_network`: Physical network name for the segments
- `segmentation_id`: Segmentation ID for the segments

Returns
A generator of segment objects

Return type
Segment

`update_segment`(*segment*, ***attrs*)

Update a segment

Parameters

- **segment** Either the id of a segment or a *Segment* instance.
- **attrs** The attributes to update on the segment represented by `segment`.

Returns
The update segment

Return type
Segment

Flavor Operations

class openstack.network.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_flavor(***attrs*)

Create a new network service flavor from attributes

Parameters

attrs Keyword arguments which will be used to create a *Flavor*, comprised of the properties on the Flavor class.

Returns

The results of flavor creation

Return type

Flavor

delete_flavor(*flavor*, *ignore_missing=True*)

Delete a network service flavor

Parameters

- **flavor** The value can be either the ID of a flavor or a *Flavor* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the flavor does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent flavor.

Returns

`None`

find_flavor(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single network service flavor

Parameters

- **name_or_id** The name or ID of a flavor.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns

One *Flavor* or None

get_flavor(*flavor*)

Get a single network service flavor

Parameters

flavor The value can be the ID of a flavor or a *Flavor* instance.

Returns

One *Flavor*

Raises
NotFoundException when no resource can be found.

update_flavor(*flavor*, ***attrs*)

Update a network service flavor

Parameters

- **flavor** Either the id of a flavor or a *Flavor* instance.
- **attrs** The attributes to update on the flavor represented by `flavor`.

Returns
The updated flavor

Return type
Flavor

flavors(***query*)

Return a generator of network service flavors

Parameters
query (*dict*) Optional query parameters to be sent to limit the resources being returned. Valid parameters include:

- `description`: The description of a flavor.
- `is_enabled`: Whether a flavor is enabled.
- `name`: The name of a flavor.
- `service_type`: The service type to which a falvor applies.

Returns
A generator of flavor objects

Return type
Flavor

Service Profile Operations

class openstack.network.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

associate_flavor_with_service_profile(*flavor*, *service_profile*)

Associate network flavor with service profile.

Parameters

- **flavor** Either the id of a flavor or a *Flavor* instance.
- **service_profile** The value can be either the ID of a service profile or a *ServiceProfile* instance.

Returns

disassociate_flavor_from_service_profile(*flavor*, *service_profile*)

Disassociate network flavor from service profile.

Parameters

- **flavor** Either the id of a flavor or a *Flavor* instance.
- **service_profile** The value can be either the ID of a service profile or a *ServiceProfile* instance.

Returns

create_service_profile(***attrs*)

Create a new network service flavor profile from attributes

Parameters

attrs Keyword arguments which will be used to create a *ServiceProfile*, comprised of the properties on the ServiceProfile class.

Returns

The results of service profile creation

Return type

ServiceProfile

delete_service_profile(*service_profile*, *ignore_missing=True*)

Delete a network service flavor profile

Parameters

- **service_profile** The value can be either the ID of a service profile or a *ServiceProfile* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the service profile does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent service profile.

Returns

None

find_service_profile(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single network service flavor profile

Parameters

- **name_or_id** The name or ID of a service profile.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns

One *ServiceProfile* or None

get_service_profile(*service_profile*)

Get a single network service flavor profile

Parameters
service_profile The value can be the ID of a service_profile or a *ServiceProfile* instance.

Returns
One *ServiceProfile*

Raises
NotFoundException when no resource can be found.

service_profiles(***query*)

Return a generator of network service flavor profiles

Parameters
query (*dict*) Optional query parameters to be sent to limit the resources returned. Available parameters inclue:

- description: The description of the service flavor profile
- driver: Provider driver for the service flavor profile
- is_enabled: Whether the profile is enabled
- project_id: The owner project ID

Returns
A generator of service profile objects

Return type
ServiceProfile

update_service_profile(*service_profile*, ***attrs*)

Update a network flavor service profile

Parameters

- **service_profile** Either the id of a service profile or a *ServiceProfile* instance.
- **attrs** The attributes to update on the service profile represented by service_profile.

Returns
The updated service profile

Return type
ServiceProfile

Tag Operations

class openstack.network.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

set_tags(*resource*, *tags*)

Replace tags of a specified resource with specified tags

Parameters

- **resource** *Resource* instance.
- **tags** (*"list"*) New tags to be set.

Returns
The updated resource

Return type
Resource

VPNaaS Operations

class openstack.network.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_vpn_endpoint_group(***attrs*)

Create a new vpn endpoint group from attributes

Parameters
attrs Keyword arguments which will be used to create a *VpnEndpointGroup*, comprised of the properties on the VpnEndpointGroup class.

Returns
The results of vpn endpoint group creation.

Return type
VpnEndpointGroup

delete_vpn_endpoint_group(*vpn_endpoint_group*, *ignore_missing=True*)

Delete a vpn service

Parameters

- **vpn_endpoint_group** The value can be either the ID of a vpn service or a *VpnEndpointGroup* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the vpn service does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent vpn service.

Returns
`None`

find_vpn_endpoint_group(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single vpn service

Parameters

- **name_or_id** The name or ID of a vpn service.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns
One *VpnEndpointGroup* or None

get_vpn_endpoint_group(*vpn_endpoint_group*)

Get a single vpn service

Parameters
vpn_endpoint_group The value can be the ID of a vpn service or a *VpnEndpointGroup* instance.

Returns
One *VpnEndpointGroup*

Raises
NotFoundException when no resource can be found.

vpn_endpoint_groups(***query*)

Return a generator of vpn services

Parameters
query (*dict*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of vpn service objects

Return type
VpnEndpointGroup

update_vpn_endpoint_group(*vpn_endpoint_group*, ***attrs*)

Update a vpn service

Parameters

- **vpn_endpoint_group** Either the id of a vpn service or a *VpnEndpointGroup* instance.
- **attrs** The attributes to update on the VPN service represented by vpn_endpoint_group.

Returns
The updated vpnservice

Return type
VpnEndpointGroup

create_vpn_ipsec_site_connection(***attrs*)

Create a new IPsec site connection from attributes

Parameters
attrs Keyword arguments which will be used to create a *VpnIPSecSiteConnection*, comprised of the properties on the IPSecSiteConnection class.

Returns
The results of IPsec site connection creation

Return type
VpnIPSecSiteConnection

find_vpn_ipsec_site_connection(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single IPsec site connection

Parameters

- **name_or_id** The name or ID of an IPsec site connection.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods such as query filters.

Returns
One *VpnIPSecSiteConnection* or None

get_vpn_ipsec_site_connection(*ipsec_site_connection*)

Get a single IPsec site connection

Parameters
ipsec_site_connection The value can be the ID of an IPsec site connection or a *VpnIPSecSiteConnection* instance.

Returns
One *VpnIPSecSiteConnection*

Raises
NotFoundException when no resource can be found.

vpn_ipsec_site_connections(***query*)

Return a generator of IPsec site connections

Parameters
query (*dict*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of IPsec site connection objects

Return type
VpnIPSecSiteConnection

update_vpn_ipsec_site_connection(*ipsec_site_connection*, ***attrs*)

Update a IPsec site connection

Ipsec_site_connection
Either the id of an IPsec site connection or a *VpnIPSecSiteConnection* instance.

Parameters
attrs The attributes to update on the IPsec site connection represented by `ipsec_site_connection`.

Returns
The updated IPsec site connection

Return type
VpnIPSecSiteConnection

delete_vpn_ipsec_site_connection(*ipsec_site_connection*, *ignore_missing=True*)

Delete a IPsec site connection

Parameters

- **ipsec_site_connection** The value can be either the ID of an IPsec site connection, or a *VpnIPSecSiteConnection* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the IPsec site connection does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent IPsec site connection.

Returns
`None`

create_vpn_ike_policy(***attrs*)

Create a new ike policy from attributes

Parameters
attrs Keyword arguments which will be used to create a *VpnIkePolicy*, comprised of the properties on the VpnIkePolicy class.

Returns
The results of ike policy creation :rtype: *VpnIkePolicy*

find_vpn_ike_policy(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single ike policy

Parameters

- **name_or_id** The name or ID of an IKE policy.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods such as query filters.

Returns
One *VpnIkePolicy* or None.

get_vpn_ike_policy(*ike_policy*)

Get a single ike policy

Parameters
ike_policy The value can be the ID of an IKE policy or a *VpnIkePolicy* instance.

Returns
One *VpnIkePolicy*

Return type
`VpnIkePolicy`

Raises
NotFoundException when no resource can be found.

vpn_ike_policies(***query*)

Return a generator of IKE policies

Parameters
query (*dict*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of ike policy objects

Return type
VpnIkePolicy

update_vpn_ike_policy(*ike_policy*, ***attrs*)

Update an IKE policy

Ike_policy
Either the IK of an IKE policy or a *VpnIkePolicy* instance.

Parameters
attrs The attributes to update on the ike policy represented by `ike_policy`.

Returns
The updated ike policy

Return type
VpnIkePolicy

delete_vpn_ike_policy(*ike_policy*, *ignore_missing=True*)

Delete an IKE policy

Parameters

- **ike_policy** The value can be either the ID of an ike policy, or a *VpnIkePolicy* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the ike policy does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent ike policy.

Returns
`None`

create_vpn_ipsec_policy(***attrs*)

Create a new IPsec policy from attributes

Parameters
attrs Keyword arguments which will be used to create a *VpnIpsecPolicy*, comprised of the properties on the VpnIpsecPolicy class.

Returns
The results of IPsec policy creation :rtype: *VpnIpsecPolicy*

find_vpn_ipsec_policy(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single IPsec policy

Parameters

- **name_or_id** The name or ID of an IPsec policy.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods such as query filters.

Returns
One *VpnIpsecPolicy* or None.

get_vpn_ipsec_policy(*ipsec_policy*)

Get a single IPsec policy

Parameters
ipsec_policy The value can be the ID of an IPcec policy or a *VpnIpsecPolicy* instance.

Returns
One *VpnIpsecPolicy*

Return type
VpnIpsecPolicy

Raises
NotFoundException when no resource can be found.

vpn_ipsec_policies(***query*)

Return a generator of IPsec policies

Parameters
query (*dict*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of IPsec policy objects

Return type
VpnIpsecPolicy

update_vpn_ipsec_policy(*ipsec_policy*, ***attrs*)

Update an IPsec policy

Ipsec_policy
Either the id of an IPsec policy or a *VpnIpsecPolicy* instance.

Parameters
attrs The attributes to update on the IPsec policy represented by `ipsec_policy`.

Returns
The updated IPsec policy

Return type
VpnIpsecPolicy

delete_vpn_ipsec_policy(*ipsec_policy*, *ignore_missing=True*)

Delete an IPsec policy

Parameters

- **ipsec_policy** The value can be either the ID of an IPsec policy, or a *VpnIpsecPolicy* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the IPsec policy does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent IPsec policy.

Returns
`None`

create_vpn_service(***attrs*)

Create a new vpn service from attributes

Parameters
attrs Keyword arguments which will be used to create a *VpnService*, comprised of the properties on the VpnService class.

Returns
The results of vpn service creation

Return type
VpnService

delete_vpn_service(*vpn_service*, *ignore_missing=True*)

Delete a vpn service

Parameters

- **vpn_service** The value can be either the ID of a vpn service or a *VpnService* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the vpn service does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent vpn service.

Returns
`None`

find_vpn_service(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single vpn service

Parameters

- **name_or_id** The name or ID of a vpn service.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns
One *VpnService* or None

get_vpn_service(*vpn_service*)

Get a single vpn service

Parameters
vpn_service The value can be the ID of a vpn service or a *VpnService* instance.

Returns
One *VpnService*

Raises
NotFoundException when no resource can be found.

vpn_services(***query*)

Return a generator of vpn services

Parameters
query (*dict*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of vpn service objects

Return type
VpnService

update_vpn_service(*vpn_service*, ***attrs*)

Update a vpn service

Parameters

- **vpn_service** Either the id of a vpn service or a *VpnService* instance.
- **attrs** The attributes to update on the VPN service represented by vpn_service.

Returns
The updated vpnservice

Return type
VpnService

Extension Operations

class openstack.network.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

find_extension(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single extension

Parameters

- **name_or_id** The name or ID of a extension.
- **ignore_missing** (*bool*) When set to False *NotFoundException* will be raised when the resource does not exist. When set to True, None will be returned when attempting to find a nonexistent resource.

- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns
One *Extension* or None

extensions(***query*)

Return a generator of extensions

Parameters
query (*dict*) Optional query parameters to be sent to limit the resources being returned. Currently no parameter is supported.

Returns
A generator of extension objects

Return type
Extension

Service Provider Operations

class openstack.network.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

service_providers(***query*)

Return a generator of service providers

Parameters
query (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of service provider objects

Return type
ServiceProvider

Local IP Operations

class openstack.network.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_local_ip(***attrs*)

Create a new local ip from attributes

Parameters
attrs Keyword arguments which will be used to create a *LocalIP*, comprised of the properties on the LocalIP class.

Returns
The results of local ip creation

Return type
LocalIP

delete_local_ip(*local_ip*, *ignore_missing=True*, *if_revision=None*)

Delete a local ip

Parameters

- **local_ip** The value can be either the ID of a local ip or a *LocalIP* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the local ip does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent ip.
- **if_revision** (*int*) Revision to put in If-Match header of update request to perform compare-and-swap update.

Returns
`None`

find_local_ip(*name_or_id*, *ignore_missing=True*, ***query*)

Find a local IP

Parameters

- **name_or_id** The name or ID of an local IP.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns
One *LocalIP* or None

get_local_ip(*local_ip*)

Get a single local ip

Parameters
local_ip The value can be the ID of a local ip or a *LocalIP* instance.

Returns
One *LocalIP*

Raises
NotFoundException when no resource can be found.

local_ips(***query*)

Return a generator of local ips

Parameters
query (*dict*) Optional query parameters to be sent to limit the resources being returned.

- `name`: Local IP name

- `description`: Local IP description
- `project_id`: Owner project ID
- `network_id`: Local IP network
- `local_port_id`: Local port ID
- `local_ip_address`: The IP address of a Local IP
- `ip_mode`: The Local IP mode

Returns
A generator of local ip objects

Return type
LocalIP

update_local_ip(*local_ip*, *if_revision=None*, ***attrs*)

Update a local ip

Parameters

- **local_ip** Either the id of a local ip or a *LocalIP* instance.
- **if_revision** (*int*) Revision to put in If-Match header of update request to perform compare-and-swap update.
- **attrs** The attributes to update on the ip represented by `value`.

Returns
The updated ip

Return type
LocalIP

create_local_ip_association(*local_ip*, ***attrs*)

Create a new local ip association from attributes

Parameters

- **local_ip** The value can be the ID of a Local IP or a *LocalIP* instance.
- **attrs** Keyword arguments which will be used to create a *LocalIPAssociation*, comprised of the properties on the LocalIP class.

Returns
The results of local ip association creation

Return type
LocalIPAssociation

delete_local_ip_association(*local_ip*, *fixed_port_id*, *ignore_missing=True*, *if_revision=None*)

Delete a local ip association

Parameters

- **local_ip** The value can be the ID of a Local IP or a *LocalIP* instance.
- **fixed_port_id** The value can be either the fixed port ID or a :class: *~openstack.network.v2.local_ip_association.LocalIPAssociation* instance.

- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the local ip association does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent ip.
- **if_revision** (*int*) Revision to put in If-Match header of update request to perform compare-and-swap update.

Returns
`None`

find_local_ip_association(*name_or_id*, *local_ip*, *ignore_missing=True*, ***query*)

Find a local ip association

Parameters

- **name_or_id** The name or ID of local ip association.
- **local_ip** The value can be the ID of a Local IP or a *LocalIP* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns
One *LocalIPAssociation* or None

get_local_ip_association(*local_ip_association*, *local_ip*)

Get a single local ip association

Parameters

- **local_ip** The value can be the ID of a Local IP or a *LocalIP* instance.
- **local_ip_association** The value can be the ID of a local ip association or a *LocalIPAssociation* instance.

Returns
One *LocalIPAssociation*

Raises
NotFoundException when no resource can be found.

local_ip_associations(*local_ip*, ***query*)

Return a generator of local ip associations

Parameters

- **local_ip** The value can be the ID of a Local IP or a *LocalIP* instance.
- **query** (*dict*) Optional query parameters to be sent to limit the resources being returned.
  - `fixed_port_id`: The ID of the port to which a local IP is associated
  - `fixed_ip`: The fixed ip address associated with a a Local IP
  - `host`: Host where local ip is associated

Returns
A generator of local ip association objects

Return type
LocalIPAssociation

Ndp Proxy Operations

class openstack.network.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_ndp_proxy(***attrs*)

Create a new ndp proxy from attributes

Parameters
attrs Keyword arguments which will be used to create a `NDPProxxy`, comprised of the properties on the NDPProxy class.

Returns
The results of ndp proxy creation

Return type
`NDPProxxy`

get_ndp_proxy(*ndp_proxy*)

Get a single ndp proxy

Parameters
ndp_proxy The value can be the ID of a ndp proxy or a *NDPProxy* instance.

Returns
One *NDPProxy*

Raises
NotFoundException when no resource can be found.

find_ndp_proxy(*ndp_proxy_id*, *ignore_missing=True*, ***query*)

Find a single ndp proxy

Parameters

- **ndp_proxy_id** The ID of a ndp proxy.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns
One *NDPProxy* or None

delete_ndp_proxy(*ndp_proxy*, *ignore_missing=True*)

Delete a ndp proxy

Parameters

- **ndp_proxy** The value can be the ID of a ndp proxy or a *NDPProxy* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the router does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent ndp proxy.

Returns
`None`

ndp_proxies(***query*)

Return a generator of ndp proxies

Parameters
query (*dict*) Optional query parameters to be sent to limit the resources being returned. Valid parameters are:

- `router_id`: The ID fo the router
- `port_id`: The ID of internal port.
- `ip_address`: The internal IP address

Returns
A generator of port forwarding objects

Return type
`PortForwarding`

update_ndp_proxy(*ndp_proxy*, ***attrs*)

Update a ndp proxy

Parameters

- **ndp_proxy** The value can be the ID of a ndp proxy or a *NDPProxy* instance.
- **attrs** The attributes to update on the ip represented by `value`.

Returns
The updated ndp_proxy

Return type
NDPProxy

BGP Operations

class openstack.network.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_bgp_peer(***attrs*)

Create a new BGP Peer from attributes

delete_bgp_peer(*peer*, *ignore_missing=True*)

Delete a BGP Peer

find_bgp_peer(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single BGP Peer

get_bgp_peer(*peer*)

Get a signle BGP Peer

update_bgp_peer(*peer*, ***attrs*)

Update a BGP Peer

bgp_peers(***query*)

Return a generator of BGP Peers

create_bgp_speaker(***attrs*)

Create a new BGP Speaker

delete_bgp_speaker(*speaker*, *ignore_missing=True*)

Delete a BGP Speaker

find_bgp_speaker(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single BGP Peer

get_bgp_speaker(*speaker*)

Get a signle BGP Speaker

update_bgp_speaker(*speaker*, ***attrs*)

Update a BGP Speaker

bgp_speakers(***query*)

Return a generator of BGP Peers

add_bgp_peer_to_speaker(*speaker*, *peer_id*)

Bind the BGP peer to the specified BGP Speaker.

remove_bgp_peer_from_speaker(*speaker*, *peer_id*)

Unbind the BGP peer from a BGP Speaker.

add_gateway_network_to_speaker(*speaker*, *network_id*)

Add a network to the specified BGP speaker.

remove_gateway_network_from_speaker(*speaker*, *network_id*)

Remove a network from the specified BGP speaker.

get_advertised_routes_of_speaker(*speaker*)

List all routes advertised by the specified BGP Speaker.

get_bgp_dragents_hosting_speaker(*speaker*)

List all BGP dynamic agents which are hosting the specified BGP Speaker.

add_bgp_speaker_to_dragent(*bgp_agent*, *bgp_speaker_id*)

Add a BGP Speaker to the specified dynamic routing agent.

get_bgp_speakers_hosted_by_dragent(*bgp_agent*)

List all BGP Seakers hosted on the specified dynamic routing agent.

remove_bgp_speaker_from_dragent(*bgp_agent*, *bgp_speaker_id*)

Delete the BGP Speaker hosted by the specified dynamic routing agent.

Tap As A Service Operations

class openstack.network.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_tap_flow(***attrs*)

Create a new Tap Flow from attributes

delete_tap_flow(*tap_flow*, *ignore_missing=True*)

Delete a Tap Flow

find_tap_flow(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single Tap Service

get_tap_flow(*tap_flow*)

Get a signle Tap Flow

update_tap_flow(*tap_flow*, ***attrs*)

Update a Tap Flow

tap_flows(***query*)

Return a generator of Tap Flows

create_tap_service(***attrs*)

Create a new Tap Service from attributes

delete_tap_service(*tap_service*, *ignore_missing=True*)

Delete a Tap Service

find_tap_service(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single Tap Service

update_tap_service(*tap_service*, ***attrs*)

Update a Tap Service

tap_services(***query*)

Return a generator of Tap Services

BGPVPN operations

class openstack.network.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_bgpvpn(***attrs*)

Create a new BGPVPN

Parameters

attrs Keyword arguments which will be used to create a *BgpVpn*, comprised of the properties on the BGPVPN class, for details see the Neutron api-ref.

Returns
The result of BGPVPN creation

Return type
BgpVpn

delete_bgpvpn(*bgpvpn*, *ignore_missing=True*)

Delete a BGPVPN

Parameters

- **bgpvpn** The value can be either the ID of a bgpvpn or a *BgpVpn* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the BGPVPN does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent BGPVPN.

Returns
`None`

find_bgpvpn(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single BGPVPN

Parameters

- **name_or_id** The name or ID of a BGPVPN.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns
One `BGPVPN` or None

get_bgpvpn(*bgpvpn*)

Get a signle BGPVPN

Parameters
bgpvpn The value can be the ID of a BGPVPN or a *BgpVpn* instance.

Returns
One *BgpVpn*

Raises
NotFoundException when no resource can be found.

update_bgpvpn(*bgppvpn*, ***attrs*)

Update a BGPVPN

Parameters

- **bgpvpn** Either the ID of a BGPVPN or a *BgpVpn* instance.
- **attrs** The attributes to update on the BGPVPN represented by `value`.

Returns
The updated BGPVPN

Return type
BgpVpn

bgpvpns(***query*)

Return a generator of BGP VPNs

Parameters
query (*dict*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of BgpVPN objects

Return type
BgpVpn

create_bgpvpn_network_association(*bgpvpn*, ***attrs*)

Create a new BGPVPN Network Association

Parameters

- **bgpvpn** The value can be either the ID of a bgpvpn or a *BgpVpn* instance.
- **attrs** Keyword arguments which will be used to create a `BgpVpnNetworkAssociation`, comprised of the properties on the BgpVpnNetworkAssociation class.

Returns
The results of BgpVpnNetworkAssociation creation

Return type
`BgpVpnNetworkAssociation`

delete_bgpvpn_network_association(*bgpvpn*, *net_association*, *ignore_missing=True*)

Delete a BGPVPN Network Association

Parameters

- **bgpvpn** The value can be either the ID of a bgpvpn or a *BgpVpn* instance.
- **net_association** The value can be either the ID of a bgpvpn_network_association or a `BgpVpnNetworkAssociation` instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the BgpVpnNetworkAssociation does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent BgpVpnNetworkAssociation.

Returns
`None`

get_bgpvpn_network_association(*bgpvpn*, *net_association*)

Get a signle BGPVPN Network Association

Parameters

- **bgpvpn** The value can be the ID of a BGPVPN or a *BgpVpn* instance.
- **net_association** The value can be the ID of a BgpVpnNetworkAssociation or a `BgpVpnNetworkAssociation` instance.

Returns
One `BgpVpnNetworkAssociation`

Raises
NotFoundException when no resource can be found.

bgpvpn_network_associations(*bgpvpn*, ***query*)

Return a generator of BGP VPN Network Associations

Param
bgpvpn: The value can be the ID of a BGPVPN or a *BgpVpn* instance.

Parameters
query (*dict*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of BgpVpnNetworkAssociation objects

Return type
`BgpVpnNetworkAssociation`

create_bgpvpn_port_association(*bgpvpn*, ***attrs*)

Create a new BGPVPN Port Association

Parameters

- **bgpvpn** The value can be either the ID of a bgpvpn or a *BgpVpn* instance.
- **attrs** Keyword arguments which will be used to create a `BgpVpnPortAssociation`, comprised of the properties on the BgpVpnPortAssociation class.

Returns
The results of BgpVpnPortAssociation creation

Return type
`BgpVpnPortAssociation`

delete_bgpvpn_port_association(*bgpvpn*, *port_association*, *ignore_missing=True*)

Delete a BGPVPN Port Association

Parameters

- **bgpvpn** The value can be either the ID of a bgpvpn or a *BgpVpn* instance.
- **port_association** The value can be either the ID of a bgpvpn_port_association or a `BgpVpnPortAssociation` instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the BgpVpnPortAssociation does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent BgpVpnPortAssociation.

Returns
`None`

find_bgpvpn_port_association(*name_or_id*, *bgpvpn_id*, *ignore_missing=True*, ***query*)

Find a single BGPVPN Port Association

Parameters

- **name_or_id** The name or ID of a BgpVpnNetworkAssociation.
- **bgpvpn_id** The value can be the ID of a BGPVPN.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns
One `BGPVPN` or None

get_bgpvpn_port_association(*bgpvpn*, *port_association*)

Get a signle BGPVPN Port Association

Parameters

- **bgpvpn** The value can be the ID of a BGPVPN or a *BgpVpn* instance.
- **port_association** The value can be the ID of a BgpVpnPortAssociation or a `BgpVpnPortAssociation` instance.

Returns
One `BgpVpnPortAssociation`

Raises
NotFoundException when no resource can be found.

update_bgpvpn_port_association(*bgpvpn*, *port_association*, ***attrs*)

Update a BPGPN Port Association

Parameters

- **bgpvpn** Either the ID of a BGPVPN or a *BgpVpn* instance.
- **port_association** The value can be the ID of a BgpVpnPortAssociation or a `BgpVpnPortAssociation` instance.
- **attrs** The attributes to update on the BGPVPN represented by `value`.

Returns
The updated BgpVpnPortAssociation.

Return type
BgpVpn

bgpvpn_port_associations(*bgpvpn*, ***query*)

Return a generator of BGP VPN Port Associations

Param
bgpvpn: The value can be the ID of a BGPVPN or a *BgpVpn* instance.

Parameters
query (*dict*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of BgpVpnNetworkAssociation objects

Return type
BgpVpnNetworkAssociation

create_bgpvpn_router_association(*bgpvpn*, ***attrs*)

Create a new BGPVPN Router Association

Parameters

- **bgpvpn** The value can be either the ID of a bgpvpn or a *BgpVpn* instance.
- **attrs** Keyword arguments which will be used to create a BgpVpnRouterAssociation, comprised of the properties on the BgpVpnRouterAssociation class.

Returns
The results of BgpVpnRouterAssociation creation

Return type
BgpVpnRouterAssociation

delete_bgpvpn_router_association(*bgpvpn*, *router_association*, *ignore_missing=True*)

Delete a BGPVPN Router Association

Parameters

- **bgpvpn** The value can be either the ID of a bgpvpn or a *BgpVpn* instance.
- **port_association** The value can be either the ID of a bgpvpn_router_association or a BgpVpnRouterAssociation instance.
- **ignore_missing** (*bool*) When set to False *NotFoundException* will be raised when the BgpVpnRouterAssociation does not exist. When set to True, no exception will be set when attempting to delete a nonexistent BgpVpnRouterAsociation.

Returns
None

get_bgpvpn_router_association(*bgpvpn*, *router_association*)

Get a signle BGPVPN Router Association

Parameters

- **bgpvpn** The value can be the ID of a BGPVPN or a *BgpVpn* instance.
- **router_association** The value can be the ID of a BgpVpnRouterAssociation or a BgpVpnRouterAssociation instance.

Returns
One BgpVpnRouterAssociation

Raises
NotFoundException when no resource can be found.

update_bgpvpn_router_association(*bgpvpn*, *router_association*, ***attrs*)

Update a BPGPN Router Association

Parameters
query (*dict*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of BgpVpnNetworkAssociation objects

Return type
`BgpVpnNetworkAssociation`

bgpvpn_router_associations(*bgpvpn*, ***query*)
Return a generator of BGP VPN router Associations

Param
bgpvpn: The value can be the ID of a BGPVPN or a *BgpVpn* instance.

Parameters
query (*dict*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of BgpVpnRouterAssociation objects

Return type
`BgpVpnRouterAssociation`

SFC operations

class openstack.network.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_sfc_flow_classifier(***attrs*)
Create a new Flow Classifier from attributes

Parameters
attrs Keyword arguments which will be used to create a *SfcFlowClassifier*, comprised of the properties on the SfcFlowClassifier class.

Returns
The results of SFC Flow Classifier creation

Return type
SfcFlowClassifier

delete_sfc_flow_classifier(*flow_classifier*, *ignore_missing=True*)
Delete a Flow Classifier

Parameters

- **flow_classifier** The value can be either the ID of a flow classifier or a *SfcFlowClassifier* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the flow classifier does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent flow classifier.

Returns
None

find_sfc_flow_classifier(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single Flow Classifier

Parameters

- **name_or_id** (*str*) The name or ID of an SFC flow classifier.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns

One `SfcFlowClassifier` or None

get_sfc_flow_classifier(*flow_classifier*)

Get a single Flow Classifier

Parameters

flow_classifier The value can be the ID of an SFC flow classifier or a *SfcFlowClassifier* instance.

Returns

SfcFlowClassifier

Raises

NotFoundException when no resource can be found.

update_sfc_flow_classifier(*flow_classifier*, ***attrs*)

Update a Flow Classifier

Parameters

- **flow_classifier** The value can be the ID of a Flow Classifier *SfcFlowClassifier*, instance.
- **attrs** The attributes to update on the Flow Classifier

Returns

The updated Flow Classifier.

Return type

SfcFlowClassifier

create_sfc_port_chain(***attrs*)

Create a new Port Chain from attributes

Parameters

attrs Keyword arguments which will be used to create a *SfcPortChain*, comprised of the properties on the SfcPortchain class.

Returns

The results of SFC Port Chain creation

Return type

SfcPortChain

delete_sfc_port_chain(*port_chain*, *ignore_missing=True*)

Delete a Port Chain

Parameters

- **port_chain** The value can be either the ID of a port chain or a *SfcPortChain* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the port chain does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent port chain.

Returns

`None`

find_sfc_port_chain(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single Port Chain

Parameters

- **name_or_id** (*str*) The name or ID of an SFC port chain.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns

One `SfcPortChain` or None

get_sfc_port_chain(*port_chain*)

Get a signle Port Chain

Parameters

port_chain The value can be the ID of an SFC port chain or a *SfcPortChain* instance.

Returns

`SfcPortchain`

Raises

NotFoundException when no resource can be found.

update_sfc_port_chain(*port_chain*, ***attrs*)

Update a Port Chain

Parameters

- **flow_classifier** The value can be the ID of a Flow Classifier *SfcFlowClassifier*, instance.
- **attrs** The attributes to update on the Flow Classifier

Returns

The updated Flow Classifier.

Return type

SfcFlowClassifier

create_sfc_port_pair(***attrs*)

Create a new Port Pair from attributes

Parameters

attrs Keyword arguments which will be used to create a *SfcPortPair*, comprised of the properties on the SfcPortPair class.

Returns

The results of SFC Port Pair creation

Return type

SfPortPair

delete_sfc_port_pair(*port_pair*, *ignore_missing=True*)

Delete a Port Pair

Parameters

- **port_pair** The value can be either the ID of a port pair or a *SfcPortPair* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the port pair does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent port pair.

Returns

None

find_sfc_port_pair(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single Port Pair

Parameters

- **name_or_id** (*str*) The name or ID of an SFC port pair.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns

One *SfcPortPair* or None

get_sfc_port_pair(*port_pair*)

Get a signle Port Pair

Parameters

port_pair The value can be the ID of an SFC port pair or a *SfcPortPair* instance.

Returns

SfcPortPair

Raises

NotFoundException when no resource can be found.

update_sfc_port_pair(*port_pair*, ***attrs*)

Update a Port Pair

Parameters

- **port_pair** The value can be the ID of a Port Pair *SfcPortPair*, instance.
- **attrs** The attributes to update on the Port Pair

Returns

The updated Port Pair.

Return type

SfcPortPair

create_sfc_port_pair_group(***attrs*)

Create a new Port Pair Group from attributes

Parameters

attrs Keyword arguments which will be used to create a *SfcPortPairGroup*, comprised of the properties on the SfcPortPairGroup class.

Returns

The results of SFC Port Pair Group creation

Return type

SfcPortPairGroup

delete_sfc_port_pair_group(*port_pair_group*, *ignore_missing=True*)

Delete a Port Pair Group

Parameters

- **port_pair_group** The value can be either the ID of a port pair group or a `SfcPortPairGroup` instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the port pair group does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent port pair group.

Returns

`None`

find_sfc_port_pair_group(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single Port Pair Group

Parameters

- **name_or_id** (*str*) The name or ID of an SFC port pair group.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns

One `SfcPortPairGroup` or None

get_sfc_port_pair_group(*port_pair_group*)

Get a signle Port Pair Group

Parameters

port_pair_group The value can be the ID of an SFC port pair group or a *SfcPortPairGroup* instance.

Returns

SfcPortPairGroup

Raises

NotFoundException when no resource can be found.

update_sfc_port_pair_group(*port_pair_group*, ***attrs*)

Update a Port Pair Group

Parameters

- **port_pair_group** The value can be the ID of a Port Pair Group SfcPortPairGroup, instance.
- **attrs** The attributes to update on the Port Pair Group

Returns

The updated Port Pair Group.

Return type

SfcPortPairGroup

create_sfc_service_graph(***attrs*)

Create a new Service Graph from attributes

Parameters

attrs Keyword arguments which will be used to create a *SfcServiceGraph*, comprised of the properties on the SfcServiceGraph class.

Returns

The results of SFC Service Graph creation

Return type

SfcServiceGraph

delete_sfc_service_graph(*service_graph*, *ignore_missing=True*)

Delete a Service Graph

Parameters

- **service_graph** The value can be either the ID of a service graph or a *SfcServiceGraph* instance.
- **ignore_missing** (*bool*) When set to False *NotFoundException* will be raised when the service graph does not exist. When set to True, no exception will be set when attempting to delete a nonexistent service graph.

Returns

None

find_sfc_service_graph(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single Service Graph

Parameters

- **name_or_id** (*str*) The name or ID of an SFC service graph.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns

One `SfcServiceGraph` or None

get_sfc_service_graph(*service_graph*)

Get a signle Service Graph

Parameters

service_graph The value can be the ID of an SFC service graph or a *SfcServiceGraph* instance.

Returns

SfcServiceGraph

Raises

NotFoundException when no resource can be found.

update_sfc_service_graph(*service_graph*, ***attrs*)

Update a Service Graph

Parameters

- **service_graph** The value can be the ID of a Service Graph *SfcServiceGraph*, instance.
- **attrs** The attributes to update on the Service Graph

Returns

The updated Service Graph.

Return type

SfcServiceGraph

Tap Mirror operations

class openstack.network.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_tap_mirror(***attrs*)

Create a new Tap Mirror from attributes

delete_tap_mirror(*tap_mirror*, *ignore_missing=True*)

Delete a Tap Mirror

find_tap_mirror(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single Tap Mirror

get_tap_mirror(*tap_mirror*)

Get a signle Tap Mirror

update_tap_mirror(*tap_mirror*, ***attrs*)

Update a Tap Mirror

tap_mirrors(***query*)

Return a generator of Tap Mirrors

Object Store API

For details on how to use this API, see *Using OpenStack Object Store*

The Object Store Class

The Object Store high-level interface is exposed as the `object_store` object on `Connection` objects.

Account Operations

class openstack.object_store.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

get_account_metadata()

Get metadata for this account.

Return type
`Account`

set_account_metadata(***metadata*)

Set metadata for this account.

Parameters
metadata (*kwargs*) Key/value pairs to be set as metadata on the container. Custom metadata can be set. Custom metadata are keys and values defined by the user.

delete_account_metadata(*keys*)

Delete metadata for this account.

Parameters
keys The keys of metadata to be deleted.

Container Operations

class openstack.object_store.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

containers(***query*)

Obtain Container objects for this account.

Parameters

query (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Return type

A generator of *Container* objects.

create_container(*name*, ***attrs*)

Create a new container from attributes

Parameters

- **container** Name of the container to create.
- **attrs** (*dict*) Keyword arguments which will be used to create a *Container*, comprised of the properties on the Container class.

Returns

The results of container creation

Return type

Container

delete_container(*container*, *ignore_missing=True*)

Delete a container

Parameters

- **container** The value can be either the name of a container or a *Container* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the container does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent server.

Returns

None

get_container_metadata(*container*)

Get metadata for a container

Parameters

container The value can be the name of a container or a *Container* instance.

Returns

One *Container*

Raises

NotFoundException when no resource can be found.

set_container_metadata(*container*, *refresh=True*, ***metadata*)

Set metadata for a container.

Parameters

- **container** The value can be the name of a container or a *Container* instance.
- **refresh** Flag to trigger refresh of container object re-fetch.
- **metadata** (*kwargs*) Key/value pairs to be set as metadata on the container. Both custom and system metadata can be set. Custom metadata are keys and values defined by the user. System metadata are keys defined by the Object Store and values defined by the user. The system metadata keys are:
  - *content_type*
  - *is_content_type_detected*
  - *versions_location*
  - *read_ACL*
  - *write_ACL*
  - *sync_to*
  - *sync_key*

delete_container_metadata(*container*, *keys*)

Delete metadata for a container.

Parameters

- **container** The value can be the ID of a container or a *Container* instance.
- **keys** The keys of metadata to be deleted.

Object Operations

class openstack.object_store.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

objects(*container*, ***query*)

Return a generator that yields the Containers objects.

Parameters

- **container** (*Container*) A container object or the name of a container that you want to retrieve objects from.
- **query** (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Return type

A generator of *Object* objects.

get_object(*obj*, *container=None*, *resp_chunk_size=1024*, *outfile=None*, *remember_content=False*)

Get the data associated with an object

Parameters

- **obj** The value can be the name of an object or a *Object* instance.
- **container** The value can be the name of a container or a *Container* instance.
- **resp_chunk_size** (*int*) chunk size of data to read. Only used if the results are being written to a file or stream is True. (optional, defaults to 1k)
- **outfile** Write the object to a file instead of returning the contents. If this option is given, body in the return tuple will be None. outfile can either be a file path given as a string, or a File like object.
- **remember_content** (*bool*) Flag whether object data should be saved as *data* property of the Object. When left as *false* and *outfile* is not defined data will not be saved and need to be fetched separately.

Returns

Instance of the *Object* objects.

Raises

NotFoundException when no resource can be found.

download_object(*obj*, *container=None*, ***attrs*)

Download the data contained inside an object.

Parameters

- **obj** The value can be the name of an object or a *Object* instance.
- **container** The value can be the name of a container or a *Container* instance.

Raises

NotFoundException when no resource can be found.

upload_object(*container*, *name*, *filename=None*, *md5=None*, *sha256=None*, *segment_size=None*, *use_slo=True*, *metadata=None*, *generate_checksums=None*, *data=None*, ***headers*)

Create a file object.

Automatically uses large-object segments if needed.

Parameters

- **container** The name of the container to store the file in. This container will be created if it does not exist already.
- **name** Name for the object within the container.
- **filename** The path to the local file whose contents will be uploaded. Mutually exclusive with data.
- **data** The content to upload to the object. Mutually exclusive with filename.

- **md5** A hexadecimal md5 of the file. (Optional), if it is known and can be passed here, it will save repeating the expensive md5 process. It is assumed to be accurate.
- **sha256** A hexadecimal sha256 of the file. (Optional) See md5.
- **segment_size** Break the uploaded object into segments of this many bytes. (Optional) SDK will attempt to discover the maximum value for this from the server if it is not specified, or will use a reasonable default.
- **headers** These will be passed through to the object creation API as HTTP Headers.
- **use_slo** If the object is large enough to need to be a Large Object, use a static rather than dynamic object. Static Objects will delete segment objects when the manifest object is deleted. (optional, defaults to True)
- **generate_checksums** Whether to generate checksums on the client side that get added to headers for later prevention of double uploads of identical data. (optional, defaults to True)
- **metadata** This dict will get changed into headers that set metadata of the object

Raises

:class:`~openstack.exceptions.SDKException` on operation error.

copy_object()

Copy an object.

delete_object(*obj*, *ignore_missing=True*, *container=None*)

Delete an object

Parameters

- **obj** The value can be either the name of an object or a *Container* instance.
- **container** The value can be the ID of a container or a *Container* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the object does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent server.

Returns

None

get_object_metadata(*obj*, *container=None*)

Get metadata for an object.

Parameters

- **obj** The value can be the name of an object or a *Object* instance.
- **container** The value can be the ID of a container or a *Container* instance.

Returns

One *Object*

Raises

NotFoundException when no resource can be found.

set_object_metadata(*obj*, *container=None*, ***metadata*)

Set metadata for an object.

Note: This method will do an extra HEAD call.

Parameters

- **obj** The value can be the name of an object or a *Object* instance.
- **container** The value can be the name of a container or a *Container* instance.
- **metadata** (*kwargs*) Key/value pairs to be set as metadata on the container. Both custom and system metadata can be set. Custom metadata are keys and values defined by the user. System metadata are keys defined by the Object Store and values defined by the user. The system metadata keys are:
  - *content_type*
  - *content_encoding*
  - *content_disposition*
  - *delete_after*
  - *delete_at*
  - *is_content_type_detected*

delete_object_metadata(*obj*, *container=None*, *keys=None*)

Delete metadata for an object.

Parameters

- **obj** The value can be the name of an object or a *Object* instance.
- **container** The value can be the ID of a container or a *Container* instance.
- **keys** The keys of metadata to be deleted.

Orchestration API

For details on how to use orchestration, see *Using OpenStack Orchestration*

The Orchestration Class

The orchestration high-level interface is available through the `orchestration` member of a *Connection* object. The `orchestration` member will only be added if the service is detected.

Stack Operations

class openstack.orchestration.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_stack(*preview=False*, ***attrs*)

Create a new stack from attributes

Parameters

- **preview** (*bool*) When `True`, a preview endpoint will be used to verify the template *Default: "False"*
- **attrs** (*dict*) Keyword arguments which will be used to create a *Stack*, comprised of the properties on the Stack class.

Returns
The results of stack creation

Return type
Stack

find_stack(*name_or_id*, *ignore_missing=True*, *resolve_outputs=True*)

Find a single stack

Parameters

- **name_or_id** The name or ID of a stack.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns
One *Stack* or None

stacks(***query*)

Return a generator of stacks

Parameters
query (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of stack objects

Return type
Stack

get_stack(*stack*, *resolve_outputs=True*)

Get a single stack

Parameters

- **stack** The value can be the ID of a stack or a *Stack* instance.
- **resolve_outputs** Whether stack should contain outputs resolved.

Returns
One *Stack*

Raises
NotFoundException when no resource can be found.

update_stack(*stack*, *, *preview=False*, ***attrs*)

Update a stack

Parameters

- **stack** The value can be the ID of a stack or a *Stack* instance.
- **attrs** (*kwargs*) The attributes to update on the stack represented by `value`.

Returns
The updated stack

Return type
Stack

Raises
NotFoundException when no resource can be found.

delete_stack(*stack*, *ignore_missing=True*)

Delete a stack

Parameters

- **stack** The value can be either the ID of a stack or a *Stack* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the stack does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent stack.

Returns
`None`

export_stack(*stack*)

Get the stack data in JSON format

Parameters
stack The value can be the ID or a name or an instance of *Stack*

Returns
A dictionary containing the stack data.

Raises
NotFoundException when no resource can be found.

get_stack_template(*stack*)

Get template used by a stack

Parameters
stack The value can be the ID of a stack or an instance of *Stack*

Returns
One object of *StackTemplate*

Raises
NotFoundException when no resource can be found.

get_stack_environment(*stack*)

Get environment used by a stack

Parameters
stack The value can be the ID of a stack or an instance of *Stack*

Returns
One object of *StackEnvironment*

Raises
NotFoundException when no resource can be found.

Stack Resource Operations

class openstack.orchestration.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

resources(*stack*, ***query*)

Return a generator of resources

Parameters

- **stack** This can be a stack object, or the name of a stack for which the resources are to be listed.
- **query** (*kwargs*) Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of resource objects if the stack exists and there are resources in it. If the stack cannot be found, an exception is thrown.

Return type
A generator of *Resource*

Raises
NotFoundException when the stack cannot be found.

Stack Action Operations

class openstack.orchestration.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

check_stack(*stack*)

Check a stacks status

Since this is an asynchronous action, the only way to check the result is to track the stacks status.

Parameters
stack The value can be either the ID of a stack or an instance of *Stack*.

Returns
None

suspend_stack(*stack*)

Suspend a stack status

Parameters
stack The value can be either the ID of a stack or an instance of *Stack*.

Returns
None

resume_stack(*stack*)

Resume a stack status

Parameters
stack The value can be either the ID of a stack or an instance of *Stack*.

Returns
None

Stack Event Operations

class openstack.orchestration.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

stack_events(*stack*, *resource_name=None*, ***attr*)

Get a stack events

Parameters

- **stack** The value can be the ID of a stack or an instance of *Stack*
- **resource_name** The name of resource. If the resource_name is not None, the base_path changes.

Returns
A generator of stack_events objects

Return type
StackEvent

Stack Template Operations

class openstack.orchestration.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

validate_template(*template*, *environment=None*, *template_url=None*, *ignore_errors=None*)

Validates a template.

Parameters

- **template** The stack template on which the validation is performed.
- **environment** A JSON environment for the stack, if provided.
- **template_url** A URI to the location containing the stack template for validation. This parameter is only required if the `template` parameter is None. This parameter is ignored if `template` is specified.
- **ignore_errors** A string containing comma separated error codes to ignore. Currently the only valid error code is 99001.

Returns
The result of template validation.

Raises
InvalidRequest if neither *template* not *template_url* is provided.

Raises
HttpException if the template fails the validation.

Software Configuration Operations

class openstack.orchestration.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_software_config(***attrs*)

Create a new software config from attributes

Parameters
attrs (*dict*) Keyword arguments which will be used to create a *SoftwareConfig*, comprised of the properties on the SoftwareConfig class.

Returns
The results of software config creation

Return type
SoftwareConfig

software_configs(***query*)

Returns a generator of software configs

Parameters
query (*dict*) Optional query parameters to be sent to limit the software configs returned.

Returns
A generator of software config objects.

Return type
SoftwareConfig

get_software_config(*software_config*)

Get details about a specific software config.

Parameters
software_config The value can be the ID of a software config or a instace of *SoftwareConfig*,

Returns
An object of type *SoftwareConfig*

delete_software_config(*software_config*, *ignore_missing=True*)

Delete a software config

Parameters

- **software_config** The value can be either the ID of a software config or an instance of *SoftwareConfig*
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the software config does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent software config.

Returns
`None`

Software Deployment Operations

class openstack.orchestration.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_software_deployment(***attrs*)

Create a new software deployment from attributes

Parameters
attrs (*dict*) Keyword arguments which will be used to create a *SoftwareDeployment*, comprised of the properties on the SoftwareDeployment class.

Returns
The results of software deployment creation

Return type
SoftwareDeployment

software_deployments(***query*)

Returns a generator of software deployments

Parameters
query (*dict*) Optional query parameters to be sent to limit the software deployments returned.

Returns
A generator of software deployment objects.

Return type
SoftwareDeployment

get_software_deployment(*software_deployment*)

Get details about a specific software deployment resource

Parameters

software_deployment The value can be the ID of a software deployment or an instace of *SoftwareDeployment*,

Returns

An object of type *SoftwareDeployment*

delete_software_deployment(*software_deployment*, *ignore_missing=True*)

Delete a software deployment

Parameters

- **software_deployment** The value can be either the ID of a software deployment or an instance of *SoftwareDeployment*
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the software deployment does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent software deployment.

Returns

`None`

update_software_deployment(*software_deployment*, ***attrs*)

Update a software deployment

Parameters

- **server** Either the ID of a software deployment or an instance of *SoftwareDeployment*
- **attrs** (*dict*) The attributes to update on the software deployment represented by `software_deployment`.

Returns

The updated software deployment

Return type

SoftwareDeployment

Placement API

The Placement Class

The placement high-level interface is available through the `placement` member of a *Connection* object. The `placement` member will only be added if the service is detected.

Resource Classes

class openstack.placement.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_resource_class(***attrs*)

Create a new resource class from attributes.

Parameters

attrs Keyword arguments which will be used to create a `ResourceClass`, comprised of the properties on the ResourceClass class.

Returns

The results of resource class creation

Return type

ResourceClass

delete_resource_class(*resource_class*, *ignore_missing=True*)

Delete a resource class

Parameters

- **resource_class** The value can be either the ID of a resource class or an *ResourceClass*, instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource class does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent resource class.

Returns

`None`

update_resource_class(*resource_class*, ***attrs*)

Update a resource class

Parameters

- **resource_class** The value can be either the ID of a resource class or an *ResourceClass*, instance.
- **attrs** The attributes to update on the resource class represented by `resource_class`.

Returns

The updated resource class

Return type

ResourceClass

get_resource_class(*resource_class*)

Get a single resource_class.

Parameters

resource_class The value can be either the ID of a resource class or an *ResourceClass*, instance.

Returns

An instance of *ResourceClass*

Raises

NotFoundException when no resource class matching the criteria could be found.

resource_classes(***query*)

Retrieve a generator of resource classs.

Parameters

query (*kwargs*) Optional query parameters to be sent to restrict the resource classs to be returned.

Returns

A generator of resource class instances.

Resource Providers

class openstack.placement.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_resource_provider(***attrs*)

Create a new resource provider from attributes.

Parameters

attrs Keyword arguments which will be used to create a *ResourceProvider*, comprised of the properties on the ResourceProvider class.

Returns

The results of resource provider creation

Return type

ResourceProvider

delete_resource_provider(*resource_provider*, *ignore_missing=True*)

Delete a resource provider

Parameters

- **resource_provider** The value can be either the ID of a resource provider or an *ResourceProvider*, instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource provider does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent resource provider.

Returns

`None`

update_resource_provider(*resource_provider*, ***attrs*)

Update a resource provider

Parameters

- **resource_provider** The value can be either the ID of a resource provider or an *ResourceProvider*, instance.
- **attrs** The attributes to update on the resource provider represented by `resource_provider`.

Returns
The updated resource provider

Return type
ResourceProvider

get_resource_provider(*resource_provider*)

Get a single resource_provider.

Parameters
resource_provider The value can be either the ID of a resource provider or an *ResourceProvider*, instance.

Returns
An instance of *ResourceProvider*

Raises
NotFoundException when no resource provider matching the criteria could be found.

find_resource_provider(*name_or_id*, *ignore_missing=True*)

Find a single resource_provider.

Parameters

- **name_or_id** The name or ID of a resource provider.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns
An instance of *ResourceProvider*

Raises
NotFoundException when no resource provider matching the criteria could be found.

resource_providers(***query*)

Retrieve a generator of resource providers.

Parameters
query (*kwargs*) Optional query parameters to be sent to restrict the resource providers to be returned.

Returns
A generator of resource provider instances.

get_resource_provider_aggregates(*resource_provider*)

Get a list of aggregates for a resource provider.

Parameters
resource_provider The value can be either the ID of a resource provider or an *ResourceProvider*, instance.

Returns
An instance of *ResourceProvider* with the `aggregates` attribute populated.

Raises

NotFoundException when no resource provider matching the criteria could be found.

set_resource_provider_aggregates(*resource_provider*, **aggregates*)

Update aggregates for a resource provider.

Parameters

- **resource_provider** The value can be either the ID of a resource provider or an *ResourceProvider*, instance.
- **aggregates** A list of aggregates. These aggregates will replace all aggregates currently present.

Returns

An instance of *ResourceProvider* with the `aggregates` attribute populated with the updated value.

Raises

NotFoundException when no resource provider matching the criteria could be found.

Resource Provider Inventories

class openstack.placement.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_resource_provider_inventory(*resource_provider*, *resource_class*, ***, *total*, ***attrs*)

Create a new resource provider inventory from attributes

Parameters

- **resource_provider** Either the ID of a resource provider or a *ResourceProvider* instance.
- **total** The actual amount of the resource that the provider can accommodate.
- **attrs** Keyword arguments which will be used to create a *ResourceProviderInventory*, comprised of the properties on the ResourceProviderInventory class.

Returns

The results of resource provider inventory creation

Return type

ResourceProviderInventory

delete_resource_provider_inventory(*resource_provider_inventory*, *resource_provider=None*, *ignore_missing=True*)

Delete a resource provider inventory

Parameters

- **resource_provider_inventory** The value can be either the ID of a resource provider or an *ResourceProviderInventory*, instance.
- **resource_provider** Either the ID of a resource provider or a *ResourceProvider* instance. This value must be specified when `resource_provider_inventory` is an ID.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource provider inventory does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent resource provider inventory.

Returns

`None`

update_resource_provider_inventory(*resource_provider_inventory*, *resource_provider=None*, ***, *resource_provider_generation=None*, ***attrs*)

Update a resource providers inventory

Parameters

- **resource_provider_inventory** The value can be either the ID of a resource provider inventory or an *ResourceProviderInventory*, instance.
- **resource_provider** Either the ID of a resource provider or a *ResourceProvider* instance. This value must be specified when `resource_provider_inventory` is an ID.

Attrs kwargs

The attributes to update on the resource provider inventory represented by `resource_provider_inventory`.

Returns

The updated resource provider inventory

Return type

ResourceProviderInventory

get_resource_provider_inventory(*resource_provider_inventory*, *resource_provider=None*)

Get a single resource_provider_inventory

Parameters

- **resource_provider_inventory** The value can be either the ID of a resource provider inventory or an *ResourceProviderInventory*, instance.
- **resource_provider** Either the ID of a resource provider or a *ResourceProvider* instance. This value must be specified when `resource_provider_inventory` is an ID.

Returns

An instance of *ResourceProviderInventory*

Raises
NotFoundException when no resource provider inventory matching the criteria could be found.

resource_provider_inventories(*resource_provider*, ***query*)

Retrieve a generator of resource provider inventories

Parameters

- **resource_provider** Either the ID of a resource provider or a *ResourceProvider* instance.
- **query** Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of resource provider inventory instances.

Traits

class openstack.placement.v1._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_trait(*name*)

Create a new trait

Parameters
name The name of the new trait

Returns
The results of trait creation

Return type
Trait

delete_trait(*trait*, *ignore_missing=True*)

Delete a trait

Parameters

- **trait** The value can be either the ID of a trait or an *Trait*, instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource provider inventory does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent resource provider inventory.

Returns
`None`

get_trait(*trait*)

Get a single trait

Parameters
trait The value can be either the ID of a trait or an *Trait*, instance.

Returns
An instance of *ResourceProviderInventory*

Raises
NotFoundException when no trait matching the criteria could be found.

`traits`(***query*)

Retrieve a generator of traits

Parameters
query Optional query parameters to be sent to limit the resources being returned.

Returns
A generator of trait objects

Shared File System API

The Shared File System Class

The high-level interface for accessing the shared file systems service API is available through the `shared_file_system` member of a `Connection` object. The `shared_file_system` member will only be added if the service is detected. `share` is an alias of the `shared_file_system` member.

Shared File System Availability Zones

Interact with Availability Zones supported by the Shared File Systems service.

`class openstack.shared_file_system.v2._proxy.Proxy`(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

`availability_zones`()

Retrieve shared file system availability zones

Returns
A generator of availability zone resources

Return type
AvailabilityZone

Shared File System Shares

Interact with Shares supported by the Shared File Systems service.

class openstack.shared_file_system.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

shares(*details=True*, ***query*)

Lists all shares with details

Parameters

query (*kwargs*) Optional query parameters to be sent to limit the shares being returned. Available parameters include:

- status: Filters by a share status
- share_server_id: The UUID of the share server.
- metadata: One or more metadata key and value pairs as a url encoded dictionary of strings.
- extra_specs: The extra specifications as a set of one or more key-value pairs.
- share_type_id: The UUID of a share type to query resources by.
- name: The user defined name of the resource to filter resources by.
- snapshot_id: The UUID of the shares base snapshot to filter the request based on.
- host: The host name of the resource to query with.
- share_network_id: The UUID of the share network to filter resources by.
- project_id: The ID of the project that owns the resource.
- is_public: A boolean query parameter that, when set to true, allows retrieving public resources that belong to all projects.
- share_group_id: The UUID of a share group to filter resource.
- export_location_id: The export location UUID that can be used to filter shares or share instances.
- export_location_path: The export location path that can be used to filter shares or share instances.
- name~: The name pattern that can be used to filter shares, share snapshots, share networks or share groups.
- description~: The description pattern that can be used to filter shares, share snapshots, share networks or share groups.
- with_count: Whether to show count in API response or not, default is False.
- limit: The maximum number of shares to return.
- offset: The offset to define start point of share or share group listing.
- sort_key: The key to sort a list of shares.

- sort_dir: The direction to sort a list of shares. A valid value is asc, or desc.

Returns
Details of shares resources

Return type
Share

find_share(*name_or_id*, *ignore_missing=True*, ***query*)

Find a single share

Parameters

- **name_or_id** The name or ID of a share.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **query** (*dict*) Any additional parameters to be passed into underlying methods. such as query filters.

Returns
One *Share* or None

get_share(*share_id*)

Lists details of a single share

Parameters
share The ID of the share to get

Returns
Details of the identified share

Return type
Share

delete_share(*share*, *ignore_missing=True*)

Deletes a single share

Parameters
share The ID of the share to delete

Returns
Result of the `delete`

Return type
`None`

update_share(*share_id*, ***attrs*)

Updates details of a single share.

Parameters

- **share** The ID of the share to update
- **attrs** (*dict*) The attributes to update on the share

Returns
the updated share

Return type
Share

`create_share`(***attrs*)

Creates a share from attributes

Returns
Details of the new share

Parameters
attrs (*dict*) Attributes which will be used to create a `Shares`, comprised of the properties on the Shares class. size and share are required to create a share.

Return type
Share

`revert_share_to_snapshot`(*share_id*, *snapshot_id*)

Reverts a share to the specified snapshot, which must be
the most recent one known to manila.

Parameters

- **share_id** The ID of the share to revert
- **snapshot_id** The ID of the snapshot to revert to

Returns
Result of the `revert`

Return type
`None`

`manage_share`(*protocol*, *export_path*, *service_host*, ***params*)

Manage a share.

Parameters

- **protocol** (*str*) The shared file systems protocol of this share.
- **export_path** (*str*) The export path formatted according to the protocol.
- **service_host** (*str*) The manage-share service host.
- **params** (*kwargs*) Optional parameters to be sent. Available parameters include: * name: The user defined name of the resource. * share_type: The name or ID of the share type to be used to create the resource. * driver_options: A set of one or more key and value pairs, as a dictionary of strings, that describe driver options. * is_public: The level of visibility for the share. * description: The user defiend description of the resource. * share_server_id: The UUID of the share server.

Returns
The share that was managed.

`unmanage_share`(*share_id*)

Unmanage the share with the given share ID.

Parameters
share_id The ID of the share to unmanage.

Returns
None

resize_share(*share_id*, *new_size*, *no_shrink=False*, *no_extend=False*, *force=False*)

Resizes a share, extending/shrinking the share as needed.

Parameters

- **share_id** The ID of the share to resize
- **new_size** The new size of the share in GiBs. If new_size is the same as the current size, then nothing is done.
- **no_shrink** (*bool*) If set to True, the given share is not shrunk, even if shrinking the share is required to get the share to the given size. This could be useful for extending shares to a minimum size, while not shrinking shares to the given size. This defaults to False.
- **no_extend** (*bool*) If set to True, the given share is not extended, even if extending the share is required to get the share to the given size. This could be useful for shrinking shares to a maximum size, while not extending smaller shares to that maximum size. This defaults to False.
- **force** (*bool*) Whether or not force should be used, in the case where the share should be extended.

Returns
None

Shared File System Storage Pools

Interact with the storage pool statistics exposed by the Shared File Systems Service.

class openstack.shared_file_system.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

storage_pools(*details=True*, ***query*)

Lists all back-end storage pools with details

Parameters
query (*kwargs*) Optional query parameters to be sent to limit the storage pools being returned. Available parameters include:

- pool_name: The pool name for the back end.
- host_name: The host name for the back end.
- backend_name: The name of the back end.
- capabilities: The capabilities for the storage back end.
- share_type: The share type name or UUID.

Returns
A generator of manila storage pool resources

Return type
StoragePool

Shared File System User Messages

View and manipulate asynchronous user messages emitted by the Shared File Systems service.

class openstack.shared_file_system.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

user_messages(***query*)
List shared file system user messages

Parameters
query (*kwargs*) Optional query parameters to be sent to limit the messages being returned. Available parameters include:

- action_id: The ID of the action during which the message was created.
- detail_id: The ID of the message detail.
- limit: The maximum number of shares to return.
- message_level: The message level.
- offset: The offset to define start point of share or share group listing.
- sort_key: The key to sort a list of messages.
- sort_dir: The direction to sort a list of shares.
- project_id: The ID of the project for which the message was created.
- request_id: The ID of the request during which the message was created.
- resource_id: The UUID of the resource for which the message was created.
- resource_type: The type of the resource for which the message was created.

Returns
A generator of user message resources

Return type
UserMessage

get_user_message(*message_id*)
List details of a single user message

Parameters
message_id The ID of the user message

Returns
Details of the identified user message

Return type
UserMessage

delete_user_message(*message_id*, *ignore_missing=True*)

Deletes a single user message

Parameters
message_id The ID of the user message

Returns
Result of the delete on the user message

Return type
UserMessage

Shared File System Limits

Get absolute limits of resources supported by the Shared File Systems service.

class openstack.shared_file_system.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

limits(***query*)

Lists all share limits.

Parameters
query (*kwargs*) Optional query parameters to be sent to limit the share limits being returned.

Returns
A generator of manila share limits resources

Return type
Limit

Shared File System Snapshots

Interact with Share Snapshots supported by the Shared File Systems service.

class openstack.shared_file_system.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

share_snapshots(*details=True*, ***query*)

Lists all share snapshots with details.

Parameters
query (*kwargs*) Optional query parameters to be sent to limit the snapshots being returned. Available parameters include:

- project_id: The ID of the user or service making the API request.

Returns
A generator of manila share snapshot resources

Return type
ShareSnapshot

`get_share_snapshot`(*snapshot_id*)

Lists details of a single share snapshot

Parameters
snapshot_id The ID of the snapshot to get

Returns
Details of the identified share snapshot

Return type
ShareSnapshot

`create_share_snapshot`(***attrs*)

Creates a share snapshot from attributes

Returns
Details of the new share snapshot

Return type
ShareSnapshot

`update_share_snapshot`(*snapshot_id*, ***attrs*)

Updates details of a single share.

Parameters
snapshot_id The ID of the snapshot to update

Pram dict attrs
The attributes to update on the snapshot

Returns
the updated share snapshot

Return type
ShareSnapshot

`delete_share_snapshot`(*snapshot_id*, *ignore_missing=True*)

Deletes a single share snapshot

Parameters
snapshot_id The ID of the snapshot to delete

Returns
Result of the `delete`

Return type
`None`

Shared File System Share Snapshot Instances

Interact with Share Snapshot Instances supported by the Shared File Systems service.

class openstack.shared_file_system.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

share_snapshot_instances(*details=True*, ***query*)

Lists all share snapshot instances with details.

Parameters

- **details** (*bool*) Whether to fetch detailed resource descriptions. Defaults to True.
- **query** (*kwargs*) Optional query parameters to be sent to limit the share snapshot instance being returned. Available parameters include:
  - **snapshot_id: The UUID of the shares base snapshot to filter** the request based on.
  - **project_id: The project ID of the user or service making the** request.

Returns

A generator of share snapshot instance resources

Return type

ShareSnapshotInstance

get_share_snapshot_instance(*snapshot_instance_id*)

Lists details of a single share snapshot instance

Parameters

snapshot_instance_id The ID of the snapshot instance to get

Returns

Details of the identified snapshot instance

Return type

ShareSnapshotInstance

Shared File System Share Networks

Create and manipulate Share Networks with the Shared File Systems service.

class openstack.shared_file_system.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

share_networks(*details=True*, ***query*)

Lists all share networks with details.

Parameters

query (*dict*) Optional query parameters to be sent to limit the resources being returned. Available parameters include:

- name~: The user defined name of the resource to filter resources by.
- project_id: The ID of the user or service making the request.
- description~: The description pattern that can be used to filter shares, share snapshots, share networks or share groups.
- all_projects: (Admin only). Defines whether to list the requested resources for all projects.

Returns

Details of shares networks

Return type

ShareNetwork

get_share_network(*share_network_id*)

Lists details of a single share network

Parameters

share_network The ID of the share network to get

Returns

Details of the identified share network

Return type

ShareNetwork

delete_share_network(*share_network_id*, *ignore_missing=True*)

Deletes a single share network

Parameters

share_network_id The ID of the share network to delete

Return type

None

update_share_network(*share_network_id*, ***attrs*)

Updates details of a single share network.

Parameters

share_network_id The ID of the share network to update

Pram dict attrs

The attributes to update on the share network

Returns

the updated share network

Return type

ShareNetwork

create_share_network(***attrs*)

Creates a share network from attributes

Returns
Details of the new share network

Parameters
attrs (*dict*) Attributes which will be used to create a ShareNetwork,comprised of the properties on the ShareNetwork class.

Return type
ShareNetwork

Shared File System Share Instances

Administrators can list, show information for, explicitly set the state of, and force-delete share instances within the Shared File Systems Service.

class openstack.shared_file_system.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

share_instances(***query*)

Lists all share instances.

Parameters
query (*kwargs*) Optional query parameters to be sent to limit the share instances being returned. Available parameters include:

- export_location_id: The export location UUID that can be used to filter share instances.
- export_location_path: The export location path that can be used to filter share instances.

Returns
Details of share instances resources

Return type
ShareInstance

get_share_instance(*share_instance_id*)

Shows details for a single share instance

Parameters
share_instance_id The UUID of the share instance to get

Returns
Details of the identified share instance

Return type
ShareInstance

reset_share_instance_status(*share_instance_id*, *status*)

Explicitly updates the state of a share instance.

Parameters

- **share_instance_id** The UUID of the share instance to reset.
- **status** The share or share instance status to be set.

Returns
None

delete_share_instance(*share_instance_id*)

Force-deletes a share instance

Parameters
share_instance The ID of the share instance to delete

Returns
None

Shared File System Share Network Subnets

Create and manipulate Share Network Subnets with the Shared File Systems service.

class openstack.shared_file_system.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

share_network_subnets(*share_network_id*)

Lists all share network subnets with details.

Parameters
share_network_id The id of the share network for which Share Network Subnets should be listed.

Returns
A generator of manila share network subnets

Return type
ShareNetworkSubnet

get_share_network_subnet(*share_network_id*, *share_network_subnet_id*)

Lists details of a single share network subnet.

Parameters

- **share_network_id** The id of the share network associated with the Share Network Subnet.
- **share_network_subnet_id** The id of the Share Network Subnet to retrieve.

Returns
Details of the identified share network subnet

Return type
ShareNetworkSubnet

create_share_network_subnet(*share_network_id*, ***attrs*)

Creates a share network subnet from attributes

Parameters

- **share_network_id** The id of the share network wthin which the the Share Network Subnet should be created.
- **attrs** (*dict*) Attributes which will be used to create a share network subnet.

Returns
Details of the new share network subnet.

Return type
ShareNetworkSubnet

delete_share_network_subnet(*share_network_id*, *share_network_subnet*, *ignore_missing=True*)

Deletes a share network subnet.

Parameters

- **share_network_id** The id of the Share Network associated with the Share Network Subnet.
- **share_network_subnet** The id of the Share Network Subnet which should be deleted.

Returns
Result of the `delete`

Return type
None

Shared File System Share Access Rules

Create, View, and Delete access rules for shares from the Shared File Systems service. Access rules can also have their deletion and visibility restricted during creation. A lock reason can also be specified. The deletion restriction can be removed during the access removal.

class openstack.shared_file_system.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

access_rules(*share*, ***query*)

Lists the access rules on a share.

Returns
A generator of the share access rules.

Return type
`ShareAccessRules`

get_access_rule(*access_id*)

List details of an access rule.

Parameters
access_id The id of the access rule to get

Returns
Details of the identified access rule.

Return type
ShareAccessRules

create_access_rule(*share_id*, ***attrs*)

Creates an access rule from attributes

Returns
Details of the new access rule

Parameters

- **share_id** The ID of the share
- **attrs** (*dict*) Attributes which will be used to create a ShareAccessRules, comprised of the properties on the ShareAccessRules class.

Return type
ShareAccessRules

delete_access_rule(*access_id*, *share_id*, *ignore_missing=True*, ***, *unrestrict=False*)

Deletes an access rule

Parameters

- **access_id** The id of the access rule to get
- **share_id** The ID of the share
- **unrestrict** If Manila must attempt removing locks while deleting

Return type
requests.models.Response HTTP response from internal requests client

Shared File System Share Groups

Interact with Share groups supported by the Shared File Systems service.

class openstack.shared_file_system.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

share_groups(***query*)

Lists all share groups.

Parameters

query (*kwargs*) Optional query parameters to be sent to limit the share groups being returned. Available parameters include:

- status: Filters by a share group status.
- **name: The user defined name of the resource to filter resources** by.
- **description: The user defined description text that can be used** to filter resources.
- project_id: The project ID of the user or service.
- share_server_id: The UUID of the share server.
- **snapshot_id: The UUID of the shares base snapshot to filter** the request based on.
- host: The host name for the back end.
- **share_network_id: The UUID of the share network to filter** resources by.
- **share_group_type_id: The share group type ID to filter** share groups.
- **share_group_snapshot_id: The source share group snapshot ID to** list the share group.
- **share_types: A list of one or more share type IDs. Allows** filtering share groups.
- limit: The maximum number of share groups members to return.
- **offset: The offset to define start point of share or share** group listing.
- sort_key: The key to sort a list of shares.
- sort_dir: The direction to sort a list of shares
- **name~: The name pattern that can be used to filter shares,** share snapshots, share networks or share groups.
- **description~: The description pattern that can be used to** filter shares, share snapshots, share networks or share groups.

Returns

A generator of manila share group resources

Return type

ShareGroup

get_share_group(*share_group_id*)

Lists details for a share group.

Parameters

share The ID of the share group to get

Returns

Details of the identified share group

Return type
ShareGroup

`find_share_group`(*name_or_id*, *ignore_missing=True*)

Finds a single share group

Parameters

- **name_or_id** The name or ID of a share group.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns
One ShareGroup or None

`create_share_group`(***attrs*)

Creates a share group from attributes

Returns
Details of the new share group

Return type
ShareGroup

`update_share_group`(*share_group_id*, ***kwargs*)

Updates details of a single share group

Parameters
share The ID of the share group

Returns
Updated details of the identified share group

Return type
ShareGroup

`delete_share_group`(*share_group_id*, *ignore_missing=True*)

Deletes a single share group

Parameters
share The ID of the share group

Returns
Result of the delete on share group

Return type
ShareGroup

Shared File System Share Group Snapshots

Interact with Share Group Snapshots by the Shared File Systems service.

class openstack.shared_file_system.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

share_group_snapshots(*details=True*, ***query*)

Lists all share group snapshots.

Parameters

query (*kwargs*) Optional query parameters to be sent to limit the share group snapshots being returned. Available parameters include:

- project_id: The ID of the project that owns the resource.
- name: The user defined name of the resource to filter resources.
- description: The user defined description text that can be used to filter resources.
- status: Filters by a share status
- share_group_id: The UUID of a share group to filter resource.
- limit: The maximum number of share group snapshot members to return.
- offset: The offset to define start point of share or share group listing.
- sort_key: The key to sort a list of shares.
- sort_dir: The direction to sort a list of shares. A valid value is asc, or desc.

Returns

Details of share group snapshots resources

Return type

ShareGroupSnapshot

get_share_group_snapshot(*group_snapshot_id*)

Show share group snapshot details

Parameters

group_snapshot_id The ID of the group snapshot to get

Returns

Details of the group snapshot

Return type

ShareGroupSnapshot

create_share_group_snapshot(*share_group_id*, ***attrs*)

Creates a point-in-time snapshot copy of a share group.

Returns

Details of the new snapshot

Parameters

- **attrs** (*dict*) Attributes which will be used to create a ShareGroupSnapshots,
- **'share_group_id'** ID of the share group to have the snapshot taken.

Return type
ShareGroupSnapshot

reset_share_group_snapshot_status(*group_snapshot_id*, *status*)

Reset share group snapshot state.

Parameters

- **group_snapshot_id** The ID of the share group snapshot to reset
- **status** The state of the share group snapshot to be set, A valid value is creating, error, available, deleting, error_deleting.

Return type
None

update_share_group_snapshot(*group_snapshot_id*, ***attrs*)

Updates a share group snapshot.

Parameters

- **group_snapshot_id** The ID of the share group snapshot to update
- **attrs** (*dict*) The attributes to update on the share group snapshot

Returns
the updated share group snapshot

Return type
ShareGroupSnapshot

delete_share_group_snapshot(*group_snapshot_id*, *ignore_missing=True*)

Deletes a share group snapshot.

Parameters
group_snapshot_id The ID of the share group snapshot to delete

Return type
None

Shared File System Share Metadata

List, Get, Create, Update, and Delete metadata for shares from the Shared File Systems service.

class openstack.shared_file_system.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

`get_share_metadata`(*share_id*)

Lists all metadata for a share.

Parameters
share_id The ID of the share

Returns
A *Share* with the shares metadata.

Return type
Share

`get_share_metadata_item`(*share_id*, *key*)

Retrieves a specific metadata item from a share by its key.

Parameters

- **share_id** The ID of the share
- **key** The key of the share metadata

Returns
A *Share* with the shares metadata.

Return type
Share

`create_share_metadata`(*share_id*, ***metadata*)

Creates share metadata as key-value pairs.

Parameters

- **share_id** The ID of the share
- **metadata** The metadata to be created

Returns
A *Share* with the shares metadata.

Return type
Share

`update_share_metadata`(*share_id*, *metadata*, *replace=False*)

Updates metadata of given share.

Parameters

- **share_id** The ID of the share
- **metadata** The metadata to be created
- **replace** Boolean for whether the preexisting metadata should be replaced

Returns
A *Share* with the shares updated metadata.

Return type
Share

`delete_share_metadata`(*share_id*, *keys*, *ignore_missing=True*)

Deletes a single metadata item on a share, idetified by its key.

Parameters

- **share_id** The ID of the share
- **keys** The list of share metadata keys to be deleted
- **ignore_missing** Boolean indicating if missing keys should be ignored.

Returns
None

Return type
None

Shared File System Resource Locks

Create, list, update and delete locks for resources. When a resource is locked, it means that it can be deleted only by services, admins or the user that created the lock.

class openstack.shared_file_system.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

resource_locks(***query*)

Lists all resource locks.

Parameters

query (*kwargs*) Optional query parameters to be sent to limit the resource locks being returned. Available parameters include:

- **project_id: The project ID of the user that the lock is** created for.
- user_id: The ID of a user to filter resource locks by.
- all_projects: list locks from all projects (Admin Only)
- **resource_id: The ID of the resource that the locks pertain to** filter resource locks by.
- **resource_action: The action prevented by the filtered resource** locks.
- **resource_type: The type of the resource that the locks pertain** to filter resource locks by.
- lock_context: The lock creators context to filter locks by.
- **lock_reason: The lock reason that can be used to filter resource** locks. (Inexact search is also available with lock_reason~)
- **created_since: Search for the list of resources that were created** after the specified date. The date is in yyyy-mm-dd format.
- **created_before: Search for the list of resources that were** created prior to the specified date. The date is in yyyy-mm-dd format.

- limit: The maximum number of resource locks to return.
- **offset: The offset to define start point of resource lock** listing.
- sort_key: The key to sort a list of shares.
- sort_dir: The direction to sort a list of shares
- **with_count: Whether to show count in API response or not,** default is False. This query parameter is useful with pagination.

Returns
A generator of manila resource locks

Return type
ResourceLock

`get_resource_lock`(*resource_lock*)

Show details of a resource lock.

Parameters
resource_lock The ID of a resource lock or a ResourceLock instance.

Returns
Details of the identified resource lock.

Return type
ResourceLock

`update_resource_lock`(*resource_lock*, ***attrs*)

Updates details of a single resource lock.

Parameters

- **resource_lock** The ID of a resource lock or a ResourceLock instance.
- **attrs** (*dict*) The attributes to update on the resource lock

Returns
the updated resource lock

Return type
ResourceLock

`delete_resource_lock`(*resource_lock*, *ignore_missing=True*)

Deletes a single resource lock

Parameters
resource_lock The ID of a resource lock or a ResourceLock instance.

Returns
Result of the delete

Return type
None

`create_resource_lock`(***attrs*)

Locks a resource.

Parameters
attrs (*dict*) Attributes which will be used to create a ResourceLock, comprised of the properties on the ResourceLock class. Available parameters include:

- resource_id: ID of the resource to be locked.
- resource_type: type of the resource (share, access_rule).
- resource_action: action to be locked (delete, show).
- **lock_reason: reason why youre locking the resource** (Optional).

Returns
Details of the lock

Return type
ResourceLock

Workflow API

The Workflow Class

The workflow high-level interface is available through the workflow member of a *Connection* object. The workflow member will only be added if the service is detected.

Workflow Operations

class openstack.workflow.v2._proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

create_workflow(***attrs*)
Create a new workflow from attributes

Parameters
attrs (*dict*) Keyword arguments which will be used to create a *Workflow*, comprised of the properties on the Workflow class.

Returns
The results of workflow creation

Return type
Workflow

update_workflow(*workflow*, ***attrs*)
Update workflow from attributes

Parameters

- **workflow** The value can be either the name of a workflow or a *Workflow* instance.
- **attrs** (*dict*) Keyword arguments which will be used to update a *Workflow*, comprised of the properties on the Workflow class.

Returns
The results of workflow update

Return type
Workflow

get_workflow(**attrs*)

Get a workflow

Parameters
workflow The value can be the name of a workflow or *Workflow* instance.

Returns
One *Workflow*

Raises
NotFoundException when no workflow matching the name could be found.

workflows(***query*)

Retrieve a generator of workflows

Parameters
query (*kwargs*) Optional query parameters to be sent to restrict the workflows to be returned. Available parameters include:

- limit: Requests at most the specified number of items be returned from the query.
- marker: Specifies the ID of the last-seen workflow. Use the limit parameter to make an initial limited request and use the ID of the last-seen workflow from the response as the marker parameter value in a subsequent limited request.

Returns
A generator of workflow instances.

delete_workflow(*value*, *ignore_missing=True*)

Delete a workflow

Parameters

- **value** The value can be either the name of a workflow or a *Workflow* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the workflow does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent workflow.

Returns
`None`

find_workflow(*name_or_id*, *ignore_missing=True*)

Find a single workflow

Parameters

- **name_or_id** The name or ID of an workflow.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns
One `Extension` or None

Execution Operations

`class openstack.workflow.v2._proxy.Proxy(session, statsd_client=None, statsd_prefix=None, prometheus_counter=None, prometheus_histogram=None, influxdb_config=None, influxdb_client=None, *args, **kwargs)`

`create_execution(**attrs)`

Create a new execution from attributes

Parameters

- **workflow_name** The name of target workflow to execute.
- **attrs** (*dict*) Keyword arguments which will be used to create a *Execution*, comprised of the properties on the Execution class.

Returns
The results of execution creation

Return type
Execution

`get_execution(*attrs)`

Get a execution

Parameters

- **workflow_name** The name of target workflow to execute.
- **execution** The value can be either the ID of a execution or a *Execution* instance.

Returns
One *Execution*

Raises
NotFoundException when no execution matching the criteria could be found.

`executions(**query)`

Retrieve a generator of executions

Parameters
query (*kwargs*) Optional query parameters to be sent to restrict the executions to be returned. Available parameters include:

- limit: Requests at most the specified number of items be returned from the query.
- marker: Specifies the ID of the last-seen execution. Use the limit parameter to make an initial limited request and use the ID of the last-seen execution from the response as the marker parameter value in a subsequent limited request.

Returns
A generator of execution instances.

`delete_execution`(*value*, *ignore_missing=True*)
Delete an execution

Parameters

- **value** The value can be either the name of a execution or a `Execution` instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the execution does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent execution.

Returns
`None`

`find_execution`(*name_or_id*, *ignore_missing=True*)
Find a single execution

Parameters

- **name_or_id** The name or ID of an execution.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.

Returns
One `Execution` or None

Cron Trigger Operations

`class openstack.workflow.v2._proxy.Proxy`(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

`create_cron_trigger`(***attrs*)
Create a new cron trigger from attributes

Parameters
attrs (*dict*) Keyword arguments which will be used to create a *CronTrigger*, comprised of the properties on the CronTrigger class.

Returns
The results of cron trigger creation

Return type
CronTrigger

`get_cron_trigger`(*cron_trigger*)
Get a cron trigger

Parameters
cron_trigger The value can be the name of a cron_trigger or *CronTrigger* instance.

Returns
One *CronTrigger*

Raises
NotFoundException when no cron triggers matching the criteria could be found.

cron_triggers(*, *all_projects=False*, ***query*)

Retrieve a generator of cron triggers

Parameters

- **all_projects** (*bool*) When set to `True`, list cron triggers from all projects. Admin-only by default.
- **query** (*kwargs*) Optional query parameters to be sent to restrict the cron triggers to be returned. Available parameters include:
  - limit: Requests at most the specified number of items be returned from the query.
  - marker: Specifies the ID of the last-seen cron trigger. Use the limit parameter to make an initial limited request and use the ID of the last-seen cron trigger from the response as the marker parameter value in a subsequent limited request.

Returns
A generator of CronTrigger instances.

delete_cron_trigger(*value*, *ignore_missing=True*)

Delete a cron trigger

Parameters

- **value** The value can be either the name of a cron trigger or a *CronTrigger* instance.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the cron trigger does not exist. When set to `True`, no exception will be set when attempting to delete a nonexistent cron trigger.

Returns
`None`

find_cron_trigger(*name_or_id*, *ignore_missing=True*, ***, *all_projects=False*, ***query*)

Find a single cron trigger

Parameters

- **name_or_id** The name or ID of a cron trigger.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **all_projects** (*bool*) When set to `True`, search for cron triggers by name across all projects. Note that this will likely result in a higher chance of duplicates.

- **query** (*kwargs*) Optional query parameters to be sent to limit the cron triggers being returned.

Returns
One `CronTrigger` or None

Raises
NotFoundException when no resource can be found.

Raises
DuplicateResource when multiple resources are found.

Resource Interface

The *Resource* layer is a lower-level interface to communicate with OpenStack services. While the classes exposed by the *Connection* build a convenience layer on top of this, *Resources* can be used directly. However, the most common usage of this layer is in receiving an object from a class in the *Connection* layer, modifying it, and sending it back into the *Connection* layer, such as to update a resource on the server.

The following services have exposed *Resource* classes.

Accelerator v2 Resources

openstack.accelerator.v2.device

The Device Class

The `Device` class inherits from *Resource*.

class openstack.accelerator.v2.device.**Device**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'device'

Singular form of key for resource.

resources_key = 'devices'

Plural form of key for resource.

base_path = '/devices'

The base part of the URI for this resource.

allow_create = False

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = False

Allow update operation for this resource.

allow_delete = False

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

created_at

The timestamp when this device was created.

hostname

The hostname of the device.

id

The ID of the device.

model

The model of the device.

std_board_info

The std board information of the device.

type

The type of the device.

updated_at

The timestamp when this device was updated.

uuid

The UUID of the device.

vendor

The vendor ID of the device.

vendor_board_info

The vendor board information of the device.

openstack.accelerator.v2.deployable

The Deployable Class

The `Deployable` class inherits from *Resource*.

class openstack.accelerator.v2.deployable.**Deployable**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.

- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'deployable'

Singular form of key for resource.

resources_key = 'deployables'

Plural form of key for resource.

base_path = '/deployables'

The base part of the URI for this resource.

allow_create = False

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = False

Allow update operation for this resource.

allow_delete = False

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_patch = True

Allow patch operation for this resource.

created_at

The timestamp when this deployable was created.

device_id

The device_id of the deployable.

id

The UUID of the deployable.

name

The name of the deployable.

num_accelerators

The num_accelerator of the deployable.

parent_id

The parent_id of the deployable.

root_id

The root_id of the deployable.

updated_at

The timestamp when this deployable was updated.

openstack.accelerator.v2.device_profile

The DeviceProfile Class

The `DeviceProfile` class inherits from *Resource*.

class openstack.accelerator.v2.device_profile.**DeviceProfile**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'device_profile'

Singular form of key for resource.

resources_key = 'device_profiles'

Plural form of key for resource.

base_path = '/device_profiles'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = False

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

created_at

The timestamp when this device_profile was created.

description

The description of the device profile

groups

The groups of the device profile

name

The name of the device profile

updated_at

The timestamp when this device_profile was updated.

uuid

The uuid of the device profile

create(*session*, *prepend_key=False*, **args*, ***kwargs*)

Create a remote resource based on this instance.

Parameters

- **session** (Adapter) The session to use for making this request.
- **prepend_key** A boolean indicating whether the resource_key should be prepended in a resource creation request. Default to True.
- **base_path** (*str*) Base part of the URI for creating resources, if different from *base_path*.
- **resource_request_key** (*str*) Overrides the usage of self.resource_key when prepending a key to the request body. Ignored if *prepend_key* is false.
- **resource_response_key** (*str*) Overrides the usage of self.resource_key when processing response bodies. Ignored if *prepend_key* is false.
- **microversion** (*str*) API version to override the negotiated one.
- **params** (*dict*) Additional params to pass.

Returns

This Resource instance.

Raises

MethodNotSupported if Resource.allow_create is not set to True.

openstack.accelerator.v2.accelerator_request

The AcceleratorRequest Class

The AcceleratorRequest class inherits from *Resource*.

class openstack.accelerator.v2.accelerator_request.**AcceleratorRequest**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of self._connection in Resource code should protect itself with a check for None.

resource_key = 'arq'

Singular form of key for resource.

resources_key = 'arqs'

Plural form of key for resource.

base_path = '/accelerator_requests'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_patch = True

Allow patch operation for binding.

attach_handle_info

The device address associated with this ARQ (if any)

attach_handle_type

The type of attach handle (e.g. PCI, mdev)

device_profile_name

The name of the device profile

device_profile_group_id

The id of the device profile group

device_rp_uuid

The UUID of the bound device RP (if any)

hostname

The host name to which ARQ is bound. (if any)

instance_uuid

The UUID of the instance associated with this ARQ (if any)

state

The state of the ARQ

uuid

The UUID of the ARQ

patch(*session*, *patch=None*, *prepend_key=True*, *has_body=True*, *retry_on_conflict=None*, *base_path=None*, ***, *microversion=None*)

Patch the remote resource.

Allows modifying the resource by providing a list of JSON patches to apply to it. The patches can use both the original (server-side) and SDK field names.

Parameters

- **session** (Adapter) The session to use for making this request.
- **patch** Additional JSON patch as a list or one patch item. If provided, it is applied on top of any changes to the current resource.
- **prepend_key** A boolean indicating whether the resource_key should be prepended in a resource update request. Default to True.
- **retry_on_conflict** (*bool*) Whether to enable retries on HTTP CONFLICT (409). Value of None leaves the *Adapter* defaults.
- **base_path** (*str*) Base part of the URI for modifying resources, if different from *base_path*.
- **microversion** (*str*) API version to override the negotiated one.

Returns

This Resource instance.

Raises

MethodNotSupported if Resource.allow_patch is not set to True.

create(*session*, *prepend_key=False*, **args*, ***kwargs*)

Create a remote resource based on this instance.

Parameters

- **session** (Adapter) The session to use for making this request.
- **prepend_key** A boolean indicating whether the resource_key should be prepended in a resource creation request. Default to True.
- **base_path** (*str*) Base part of the URI for creating resources, if different from *base_path*.
- **resource_request_key** (*str*) Overrides the usage of self.resource_key when prepending a key to the request body. Ignored if *prepend_key* is false.
- **resource_response_key** (*str*) Overrides the usage of self.resource_key when processing response bodies. Ignored if *prepend_key* is false.
- **microversion** (*str*) API version to override the negotiated one.
- **params** (*dict*) Additional params to pass.

Returns

This Resource instance.

Raises

MethodNotSupported if Resource.allow_create is not set to True.

Baremetal Resources

openstack.baremetal.v1.driver

The Driver Class

The Driver class inherits from *Resource*.

class openstack.baremetal.v1.driver.**Driver**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'drivers'

Plural form of key for resource.

base_path = '/drivers'

The base part of the URI for this resource.

allow_create = False

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = False

Allow update operation for this resource.

allow_delete = False

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

hosts

A list of active hosts that support this driver.

links

A list of relative links, including the self and bookmark links.

name

The name of the driver

properties

A list of links to driver properties.

default_bios_interface

Default BIOS interface implementation. Introduced in API microversion 1.40.

default_boot_interface

Default boot interface implementation. Introduced in API microversion 1.30.

default_console_interface

Default console interface implementation. Introduced in API microversion 1.30.

default_deploy_interface

Default deploy interface implementation. Introduced in API microversion 1.30.

default_firmware_interface

Default firmware interface implementation. Introduced in API microversion 1.86.

default_inspect_interface

Default inspect interface implementation. Introduced in API microversion 1.30.

default_management_interface

Default management interface implementation. Introduced in API microversion 1.30.

default_network_interface

Default network interface implementation. Introduced in API microversion 1.30.

default_power_interface

Default port interface implementation. Introduced in API microversion 1.30.

default_raid_interface

Default RAID interface implementation. Introduced in API microversion 1.30.

default_rescue_interface

Default rescue interface implementation. Introduced in API microversion 1.38.

default_storage_interface

Default storage interface implementation. Introduced in API microversion 1.33.

default_vendor_interface

Default vendor interface implementation. Introduced in API microversion 1.30.

enabled_bios_interfaces

Enabled BIOS interface implementations. Introduced in API microversion 1.40.

enabled_boot_interfaces

Enabled boot interface implementations. Introduced in API microversion 1.30.

enabled_console_interfaces

Enabled console interface implementations. Introduced in API microversion 1.30.

enabled_deploy_interfaces

Enabled deploy interface implementations. Introduced in API microversion 1.30.

enabled_firmware_interfaces

Enabled firmware interface implementations. Introduced in API microversion 1.86.

enabled_inspect_interfaces

Enabled inspect interface implementations. Introduced in API microversion 1.30.

enabled_management_interfaces

Enabled management interface implementations. Introduced in API microversion 1.30.

enabled_network_interfaces

Enabled network interface implementations. Introduced in API microversion 1.30.

enabled_power_interfaces

Enabled port interface implementations. Introduced in API microversion 1.30.

enabled_raid_interfaces

Enabled RAID interface implementations. Introduced in API microversion 1.30.

enabled_rescue_interfaces

Enabled rescue interface implementations. Introduced in API microversion 1.38.

enabled_storage_interfaces

Enabled storage interface implementations. Introduced in API microversion 1.33.

enabled_vendor_interfaces

Enabled vendor interface implementations. Introduced in API microversion 1.30.

list_vendor_passthru(*session*)

Fetch vendor specific methods exposed by driver

Parameters

session The session to use for making this request.

Returns

A dict of the available vendor passthru methods for driver. Method names keys and corresponding usages in dict form as values Usage dict properties: * `async`: bool # Is passthru function invoked asynchronously * `attach`: bool # Is return value attached to response object * `description`: str # Description of what the method does * `http_methods`: list # List of HTTP methods supported

call_vendor_passthru(*session*, *verb*, *method*, *body=None*)

Call a vendor specific passthru method

Contents of body are params passed to the hardware driver function. Validation happens there. Missing parameters, or excess parameters will cause the request to be rejected

Parameters

- **session** The session to use for making this request.
- **method** Vendor passthru method name.
- **verb** One of GET, POST, PUT, DELETE, depending on the driver and method.
- **body** passed to the vendor function as json body.

Raises

`ValueError` if `verb` is not one of GET, POST, PUT, DELETE

Returns

response of method call.

openstack.baremetal.v1.chassis

The Chassis Class

The `Chassis` class inherits from *Resource*.

class openstack.baremetal.v1.chassis.**Chassis**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'chassis'

Plural form of key for resource.

base_path = '/chassis'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_patch = True

Allow patch operation for this resource.

commit_method = 'PATCH'

Method for committing a resource (PUT, PATCH, POST)

commit_jsonpatch = True

Whether commit uses JSON patch format.

created_at

Timestamp at which the chassis was created.

description

A descriptive text about the service

extra

A set of one or more arbitrary metadata key and value pairs.

id

The UUID for the chassis

links

A list of relative links, including the self and bookmark links.

nodes

Links to the collection of nodes contained in the chassis

updated_at

Timestamp at which the chassis was last updated.

openstack.baremetal.v1.Node

The Node Class

The Node class inherits from *Resource*.

class openstack.baremetal.v1.node.**Node**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of self._connection in Resource code should protect itself with a check for None.

resources_key = 'nodes'

Plural form of key for resource.

base_path = '/nodes'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_patch = True

Allow patch operation for this resource.

commit_method = 'PATCH'

Method for committing a resource (PUT, PATCH, POST)

commit_jsonpatch = True

Whether commit uses JSON patch format.

allocation_id

The UUID of the allocation associated with this node. Added in API microversion 1.52.

owner

A string or UUID of the tenant who owns the baremetal node. Added in API microversion 1.50.

boot_mode

The current boot mode state (uefi/bios). Added in API microversion 1.75.

chassis_id

The UUID of the chassis associated wit this node. Can be empty or None.

clean_step

The current clean step.

conductor_group

Conductor group this node is managed by. Added in API microversion 1.46.

created_at

Timestamp at which the node was last updated.

deploy_step

The current deploy step. Added in API microversion 1.44.

description

The description of the node. Added in API microversion 1.51.

driver

The name of the driver.

driver_info

All the metadata required by the driver to manage this node. List of fields varies between drivers, and can be retrieved from the *openstack.baremetal.v1.driver.Driver* resource.

driver_internal_info

Internal metadata set and stored by nodes driver. This is read-only.

extra

A set of one or more arbitrary metadata key and value pairs.

fault

Fault type that caused the node to enter maintenance mode. Introduced in API microversion 1.42.

id

The UUID of the node resource.

instance_info

Information used to customize the deployed image, e.g. size of root partition, config drive in the form of base64 encoded string and other metadata.

instance_id

UUID of the nova instance associated with this node.

is_automated_clean_enabled

Override enabling of automated cleaning. Added in API microversion 1.47.

is_console_enabled

Whether console access is enabled on this node.

is_maintenance

Whether node is currently in maintenance mode. Nodes put into maintenance mode are removed from the available resource pool.

is_retired

Whether the node is marked for retirement. Added in API microversion 1.61.

is_secure_boot

Whether the node is currently booted with secure boot turned on. Added in API microversion 1.75.

last_error

Any error from the most recent transaction that started but failed to finish.

lessee

Field indicating if the node is leased to a specific project. Added in API version 1.65

links

A list of relative links, including self and bookmark links.

maintenance_reason

user settable description of the reason why the node was placed into maintenance mode.

name

Human readable identifier for the node. May be undefined. Certain words are reserved. Added in API microversion 1.5

parent_node

The node which serves as the parent_node for this node. Added in API version 1.83

ports

Links to the collection of ports on this node.

port_groups

Links to the collection of portgroups on this node. Available since API microversion 1.24.

power_state

The current power state. Usually power on or power off, but may be None if service is unable to determine the power state.

properties

Physical characteristics of the node. Content populated by the service during inspection.

provision_state

The current provisioning state of the node.

retired_reason

The reason why the node is marked for retirement. Added in API microversion 1.61.

raid_config

The current RAID configuration of the node.

reservation

The name of an service conductor host which is holding a lock on this node, if a lock is held.

resource_class

A string to be used by external schedulers to identify this node as a unit of a specific type of resource. Added in API microversion 1.21.

service_step

A string representing the current service step being executed upon. Added in API microversion 1.87.

runbook

A string representing the uuid or logical name of a runbook as an alternative to providing `clean_steps` or `service_steps`. Added in API microversion 1.92.

shard

A string indicating the shard this node belongs to. Added in API microversion 1,82.

states

Links to the collection of states.

target_provision_state

The requested state if a provisioning action has been requested. For example, `AVAILABLE`, `DEPLOYING`, `DEPLOYWAIT`, `DEPLOYING`, `ACTIVE` etc.

target_power_state

The requested state during a state transition.

target_raid_config

The requested RAID configuration of the node which will be applied when the node next transitions through the CLEANING state.

traits

Traits of the node. Introduced in API microversion 1.37.

updated_at

Timestamp at which the node was last updated.

bios_interface

BIOS interface to use when setting BIOS properties of the node. Introduced in API microversion 1.40.

boot_interface

Boot interface to use when configuring boot of the node. Introduced in API microversion 1.31.

console_interface

Console interface to use when working with serial console. Introduced in API microversion 1.31.

deploy_interface

Deploy interface to use when deploying the node. Introduced in API microversion 1.31.

firmware_interface

Firmware interface to be used when managing the node. Introduced in API microversion 1.86

inspect_interface

Inspect interface to use when inspecting the node. Introduced in API microversion 1.31.

management_interface

Management interface to use for management actions on the node. Introduced in API microversion 1.31.

network_interface

Network interface provider to use when plumbing the network connections for this node. Introduced in API microversion 1.20.

power_interface

Power interface to use for power actions on the node. Introduced in API microversion 1.31.

raid_interface

RAID interface to use for configuring RAID on the node. Introduced in API microversion 1.31.

rescue_interface

Rescue interface to use for rescuing of the node. Introduced in API microversion 1.38.

storage_interface

Storage interface to use when attaching remote storage. Introduced in API microversion 1.33.

vendor_interface

Vendor interface to use for vendor-specific actions on the node. Introduced in API microversion 1.31.

create(*session*, **args*, ***kwargs*)

Create a remote resource based on this instance.

The overridden version is capable of handling the populated `provision_state` field of one of three values: `enroll`, `manageable` or `available`. If not provided, the server default is used (`enroll` in newer versions of Ironic).

This call does not cause a node to go through automated cleaning. If you need it, use `provision_state=manageable` followed by a call to *set_provision_state()*.

Note that Bare Metal API 1.4 is required for `manageable` and 1.11 is required for `enroll`.

> **Warning**
>
> Using `provision_state=available` is only possible with API versions 1.1 to 1.10 and thus is incompatible with setting any fields that appeared after 1.11.

Parameters

session (`Adapter`) The session to use for making this request.

Returns

This `Resource` instance.

Raises

ValueError if the Nodes `provision_state` is not one of `None`, `enroll`, `manageable` or `available`.

Raises

NotSupported if the `provision_state` cannot be reached with any API version supported by the server.

commit(*session*, **args*, ***kwargs*)

Commit the state of the instance to the remote resource.

Parameters

session (`Adapter`) The session to use for making this request.

Returns

This *Node* instance.

set_provision_state(*session*, *target*, *config_drive=None*, *clean_steps=None*, *rescue_password=None*, *wait=False*, *timeout=None*, *deploy_steps=None*, *service_steps=None*, *runbook=None*)

Run an action modifying this nodes provision state.

This call is asynchronous, it will return success as soon as the Bare Metal service acknowledges the request.

Parameters

- **session** (`Adapter`) The session to use for making this request.
- **target** Provisioning action, e.g. `active`, `provide`. See the Bare Metal service documentation for available actions.
- **config_drive** Config drive to pass to the node, only valid for `active` and ``rebuild` targets. You can use functions from `openstack.baremetal.configdrive` to build it.
- **clean_steps** Clean steps to execute, only valid for `clean` target.
- **rescue_password** Password for the rescue operation, only valid for `rescue` target.
- **wait** Whether to wait for the target state to be reached.
- **timeout** Timeout (in seconds) to wait for the target state to be reached. If `None`, wait without timeout.
- **deploy_steps** Deploy steps to execute, only valid for `active` and `rebuild` target.
- **service_steps** Service steps to execute, only valid for `service` target.
- **runbook** UUID or logical name of a runbook.

Returns

This *Node* instance.

Raises

ValueError if `config_drive`, `clean_steps`, `deploy_steps` or `rescue_password` are provided with an invalid `target`.

Raises

ResourceFailure if the node reaches an error state while waiting for the state.

Raises
ResourceTimeout if timeout is reached while waiting for the state.

wait_for_power_state(*session*, *expected_state*, *timeout=None*)

Wait for the node to reach the expected power state.

Parameters

- **session** (Adapter) The session to use for making this request.
- **expected_state** The expected power state to reach.
- **timeout** If `wait` is set to `True`, specifies how much (in seconds) to wait for the expected state to be reached. The value of `None` (the default) means no client-side timeout.

Returns
This *Node* instance.

Raises
ResourceTimeout on timeout.

wait_for_provision_state(*session*, *expected_state*, *timeout=None*, *abort_on_failed_state=True*)

Wait for the node to reach the expected state.

Parameters

- **session** (Adapter) The session to use for making this request.
- **expected_state** The expected provisioning state to reach.
- **timeout** If `wait` is set to `True`, specifies how much (in seconds) to wait for the expected state to be reached. The value of `None` (the default) means no client-side timeout.
- **abort_on_failed_state** If `True` (the default), abort waiting if the node reaches a failure state which does not match the expected one. Note that the failure state for `enroll` -> `manageable` transition is `enroll` again.

Returns
This *Node* instance.

Raises
ResourceFailure if the node reaches an error state and `abort_on_failed_state` is `True`.

Raises
ResourceTimeout on timeout.

wait_for_reservation(*session*, *timeout=None*)

Wait for a lock on the node to be released.

Bare metal nodes in ironic have a reservation lock that is used to represent that a conductor has locked the node while performing some sort of action, such as changing configuration as a result of a machine state change.

This lock can occur during power syncronization, and prevents updates to objects attached to the node, such as ports.

Note that nothing prevents a conductor from acquiring the lock again after this call returns, so it should be treated as best effort.

Returns immediately if there is no reservation on the node.

Parameters

- **session** (Adapter) The session to use for making this request.
- **timeout** How much (in seconds) to wait for the lock to be released. The value of None (the default) means no timeout.

Returns
This *Node* instance.

inject_nmi(*session*)

Inject NMI.

Parameters
session The session to use for making this request.

Returns
None

set_power_state(*session*, *target*, *wait=False*, *timeout=None*)

Run an action modifying this nodes power state.

This call is asynchronous, it will return success as soon as the Bare Metal service acknowledges the request.

Parameters

- **session** (Adapter) The session to use for making this request.
- **target** Target power state, as a *PowerAction* or a string.
- **wait** Whether to wait for the expected power state to be reached.
- **timeout** Timeout (in seconds) to wait for the target state to be reached. If None, wait without timeout.

attach_vif(*session*, *vif_id*, *retry_on_conflict=True*, ***, *port_id=None*, *port_group_id=None*)

Attach a VIF to the node.

The exact form of the VIF ID depends on the network interface used by the node. In the most common case it is a Network service port (NOT a Bare Metal port) ID. A VIF can only be attached to one node at a time.

Parameters

- **session** (Adapter) The session to use for making this request.
- **vif_id** Backend-specific VIF ID.
- **retry_on_conflict** Whether to retry HTTP CONFLICT errors. This can happen when either the VIF is already used on a node or the node is locked. Since the latter happens more often, the default value is True.
- **port_id** The UUID of the port to attach the VIF to. Only one of port_id or port_group_id can be provided.

- **port_group_id** The UUID of the portgroup to attach to. Only one of port_group_id or port_id can be provided.

Returns
None

Raises
NotSupported if the server does not support the VIF API.

Raises
InvalidRequest if both port_id and port_group_id are provided.

detach_vif(*session*, *vif_id*, *ignore_missing=True*)

Detach a VIF from the node.

The exact form of the VIF ID depends on the network interface used by the node. In the most common case it is a Network service port (NOT a Bare Metal port) ID.

Parameters

- **session** (Adapter) The session to use for making this request.
- **vif_id** (*string*) Backend-specific VIF ID.
- **ignore_missing** (*bool*) When set to False *NotFoundException* will be raised when the VIF does not exist. Otherwise, False is returned.

Returns
True if the VIF was detached, otherwise False.

Raises
NotSupported if the server does not support the VIF API.

list_vifs(*session*)

List IDs of VIFs attached to the node.

The exact form of the VIF ID depends on the network interface used by the node. In the most common case it is a Network service port (NOT a Bare Metal port) ID.

Parameters
session (Adapter) The session to use for making this request.

Returns
List of VIF IDs as strings.

Raises
NotSupported if the server does not support the VIF API.

validate(*session*, *required=('boot', 'deploy', 'power')*)

Validate required information on the node.

Parameters

- **session** (Adapter) The session to use for making this request.
- **required** List of interfaces that are required to pass validation. The default value is the list of minimum required interfaces for provisioning.

Returns
dict mapping interface names to *ValidationResult* objects.

Raises
ValidationException if validation fails for a required interface.

set_maintenance(*session*, *reason=None*)

Enable maintenance mode on the node.

Parameters

- **session** (Adapter) The session to use for making this request.
- **reason** Optional reason for maintenance.

Returns
This *Node* instance.

unset_maintenance(*session*)

Disable maintenance mode on the node.

Parameters
session (Adapter) The session to use for making this request.

Returns
This *Node* instance.

get_boot_device(*session*)

Get node boot device.

Parameters
session The session to use for making this request.

Returns
The HTTP response.

set_boot_device(*session*, *boot_device*, *persistent=False*)

Set node boot device

Parameters

- **session** The session to use for making this request.
- **boot_device** Boot device to assign to the node.
- **persistent** If the boot device change is maintained after node reboot

Returns
None

get_supported_boot_devices(*session*)

Get supported boot devices for the node.

Parameters
session The session to use for making this request.

Returns
The HTTP response.

set_boot_mode(*session*, *target*)

Make a request to change nodes boot mode

This call is asynchronous, it will return success as soon as the Bare Metal service acknowledges the request.

Parameters

- **session** The session to use for making this request.
- **target** Boot mode to set for node, one of either uefi/bios.

Returns
None

Raises
ValueError if `target` is not one of uefi or bios.

set_secure_boot(*session*, *target*)

Make a request to change nodes secure boot state

This call is asynchronous, it will return success as soon as the Bare Metal service acknowledges the request.

Parameters

- **session** The session to use for making this request.
- **target** (*bool*) Boolean indicating secure boot state to set. True/False corresponding to on/off respectively.

Returns
None

Raises
ValueError if `target` is not boolean.

add_trait(*session*, *trait*)

Add a trait to the node.

Parameters

- **session** The session to use for making this request.
- **trait** The trait to add to the node.

Returns
None

remove_trait(*session*, *trait*, *ignore_missing=True*)

Remove a trait from the node.

Parameters

- **session** The session to use for making this request.
- **trait** The trait to remove from the node.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the trait does not exist. Otherwise, `False` is returned.

Returns bool
True on success removing the trait. False when the trait does not exist already.

set_traits(*session*, *traits*)

Set traits for the node.

Removes any existing traits and adds the traits passed in to this method.

Parameters

- **session** The session to use for making this request.
- **traits** list of traits to add to the node.

Returns
None

`call_vendor_passthru`(*session*, *verb*, *method*, *body=None*)

Call a vendor passthru method.

Parameters

- **session** The session to use for making this request.
- **verb** The HTTP verb, one of GET, SET, POST, DELETE.
- **method** The method to call using vendor_passthru.
- **body** The JSON body in the HTTP call.

Returns
The HTTP response.

`list_vendor_passthru`(*session*)

List vendor passthru methods for the node.

Parameters
session The session to use for making this request.

Returns
The HTTP response.

`get_console`(*session*)

Get the node console.

Parameters
session The session to use for making this request.

Returns
The HTTP response.

`set_console_mode`(*session*, *enabled*)

Set the node console mode.

Parameters

- **session** The session to use for making this request.
- **enabled** Whether the console should be enabled or not.

Returns
None

`get_node_inventory`(*session*, *node_id*)

Get a nodes inventory.

Parameters

- **session** The session to use for making this request.
- **node_id** **DEPRECATED** The ID of the node.

Returns
The HTTP response.

`list_firmware`(*session*)

List firmware components associated with the node.

Parameters
session The session to use for making this request.

Returns
The HTTP response.

`patch`(*session*, *patch=None*, *prepend_key=True*, *has_body=True*, *retry_on_conflict=None*, *base_path=None*, ***, *microversion=None*, *reset_interfaces=None*)

Patch the remote resource.

Allows modifying the resource by providing a list of JSON patches to apply to it. The patches can use both the original (server-side) and SDK field names.

Parameters

- **session** (`Adapter`) The session to use for making this request.
- **patch** Additional JSON patch as a list or one patch item. If provided, it is applied on top of any changes to the current resource.
- **prepend_key** A boolean indicating whether the resource_key should be prepended in a resource update request. Default to True.
- **retry_on_conflict** (*bool*) Whether to enable retries on HTTP CONFLICT (409). Value of `None` leaves the *Adapter* defaults.
- **base_path** (*str*) Base part of the URI for modifying resources, if different from *base_path*.
- **microversion** (*str*) API version to override the negotiated one.

Returns
This `Resource` instance.

Raises
MethodNotSupported if `Resource.allow_patch` is not set to `True`.

The PowerAction Class

The `PowerAction` enumeration represents known power actions.

`class` openstack.baremetal.v1.node.`PowerAction`(*value*)

Mapping from an action to a target power state.

`POWER_ON = 'power on'`

Power on the node.

`POWER_OFF = 'power off'`

Power off the node (using hard power off).

`REBOOT = 'rebooting'`

Reboot the node (using hard power off).

SOFT_POWER_OFF = 'soft power off'

Power off the node using soft power off.

SOFT_REBOOT = 'soft rebooting'

Reboot the node using soft power off.

The ValidationResult Class

The `ValidationResult` class represents the result of a validation.

class openstack.baremetal.v1.node.**ValidationResult**(*result*, *reason*)

Result of a single interface validation.

Variables

- **result** Result of a validation, `True` for success, `False` for failure, `None` for unsupported interface.
- **reason** If `result` is `False` or `None`, explanation of the result.

The WaitResult Class

The `WaitResult` class represents the result of waiting for several nodes.

class openstack.baremetal.v1.node.**WaitResult**(*success*, *failure*, *timeout*)

A named tuple representing a result of waiting for several nodes.

Each component is a list of *Node* objects:

Variables

- **~.success** a list of *Node* objects that reached the state.
- **~.timeout** a list of *Node* objects that reached timeout.
- **~.failure** a list of *Node* objects that hit a failure.

Create new instance of WaitResult(success, failure, timeout)

openstack.baremetal.v1.port

The Port Class

The `Port` class inherits from *Resource*.

class openstack.baremetal.v1.port.**Port**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'ports'

Plural form of key for resource.

base_path = '/ports'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_patch = True

Allow patch operation for this resource.

commit_method = 'PATCH'

Method for committing a resource (PUT, PATCH, POST)

commit_jsonpatch = True

Whether commit uses JSON patch format.

address

The physical hardware address of the network port, typically the hardware MAC address.

created_at

Timestamp at which the port was created.

extra

A set of one or more arbitrary metadata key and value pairs.

id

The UUID of the port

internal_info

Internal metadata set and stored by the port. This field is read-only. Added in API microversion 1.18.

is_pxe_enabled

Whether PXE is enabled on the port. Added in API microversion 1.19.

is_smartnic

Whether the port is a Smart NIC port. Added in API microversion 1.53.

links

A list of relative links, including the self and bookmark links.

local_link_connection

The port bindig profile. If specified, must contain `switch_id` and `port_id` fields. `switch_info` field is an optional string field to be used to store vendor specific information. Added in API microversion 1.19.

name

The name of the port

node_id

The UUID of node this port belongs to

physical_network

The name of physical network this port is attached to. Added in API microversion 1.34.

port_group_id

The UUID of PortGroup this port belongs to. Added in API microversion 1.24.

updated_at

Timestamp at which the port was last updated.

openstack.baremetal.v1.port_group

The PortGroup Class

The `PortGroup` class inherits from *Resource*.

class openstack.baremetal.v1.port_group.**PortGroup**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'portgroups'

Plural form of key for resource.

base_path = '/portgroups'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_patch = True

Allow patch operation for this resource.

commit_method = 'PATCH'

Method for committing a resource (PUT, PATCH, POST)

commit_jsonpatch = True

Whether commit uses JSON patch format.

address

The physical hardware address of the portgroup, typically the hardware MAC address. Added in API microversion 1.23.

created_at

Timestamp at which the portgroup was created.

extra

A set of one or more arbitrary metadata key and value pairs.

name

The name of the portgroup

id

The UUID for the portgroup

internal_info

Internal metadaa set and stored by the portgroup.

is_standalone_ports_supported

Whether ports that are members of this portgroup can be used as standalone ports. Added in API microversion 1.23.

links

A list of relative links, including the self and bookmark links.

mode

Port bonding mode. Added in API microversion 1.26.

node_id

UUID of the node this portgroup belongs to.

ports

A list of links to the collection of ports belonging to this portgroup. Added in API microversion 1.24.

properties

Port group properties. Added in API microversion 1.26.

updated_at

Timestamp at which the portgroup was last updated.

openstack.baremetal.v1.Allocation

The Allocation Class

The `Allocation` class inherits from *Resource*.

class openstack.baremetal.v1.allocation.**Allocation**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'allocations'

Plural form of key for resource.

base_path = '/allocations'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_patch = True

Allow patch operation for this resource.

commit_method = 'PATCH'

Method for committing a resource (PUT, PATCH, POST)

commit_jsonpatch = True

Whether commit uses JSON patch format.

candidate_nodes

The candidate nodes for this allocation.

created_at

Timestamp at which the allocation was created.

extra
A set of one or more arbitrary metadata key and value pairs.

id
The UUID for the allocation.

last_error
The last error for the allocation.

links
A list of relative links, including the self and bookmark links.

name
The name of the allocation.

node
The node UUID or name to create the allocation against, bypassing the normal allocation process.

node_id
UUID of the node this allocation belongs to.

owner
The tenant who owns the object

resource_class
The requested resource class.

state
The state of the allocation.

traits
The requested traits.

updated_at
Timestamp at which the allocation was last updated.

wait(*session*, *timeout=None*, *ignore_error=False*)
Wait for the allocation to become active.

Parameters

- **session** (Adapter) The session to use for making this request.
- **timeout** How much (in seconds) to wait for the allocation. The value of None (the default) means no client-side timeout.
- **ignore_error** If True, this call will raise an exception if the allocation reaches the error state. Otherwise the error state is considered successful and the call returns.

Returns
This *Allocation* instance.

Raises
ResourceFailure if allocation fails and ignore_error is False.

Raises
ResourceTimeout on timeout.

openstack.baremetal.v1.volume_connector

The VolumeConnector Class

The `VolumeConnector` class inherits from *Resource*.

class openstack.baremetal.v1.volume_connector.**VolumeConnector**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'connectors'

Plural form of key for resource.

base_path = '/volume/connectors'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_patch = True

Allow patch operation for this resource.

commit_method = 'PATCH'

Method for committing a resource (PUT, PATCH, POST)

commit_jsonpatch = True

Whether commit uses JSON patch format.

created_at

Timestamp at which the port was created.

extra

A set of one or more arbitrary metadata key and value pairs.

links

A list of relative links, including the self and bookmark links.

node_id

The UUID of node this port belongs to

type

The types of Volume connector

updated_at

Timestamp at which the port was last updated.

id

The UUID of the port

openstack.baremetal.v1.volume_target

The VolumeTarget Class

The `VolumeTarget` class inherits from *Resource*.

class openstack.baremetal.v1.volume_target.**VolumeTarget**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'targets'

Plural form of key for resource.

base_path = '/volume/targets'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_patch = True

Allow patch operation for this resource.

commit_method = 'PATCH'

Method for committing a resource (PUT, PATCH, POST)

commit_jsonpatch = True

Whether commit uses JSON patch format.

created_at

Timestamp at which the port was created.

extra

A set of one or more arbitrary metadata key and value pairs.

links

A list of relative links. Includes the self and bookmark links.

node_id

The UUID of the Node this resource belongs to.

properties

A set of physical information of the volume.

updated_at

Timestamp at which the port was last updated.

id

The UUID of the resource.

volume_id

The identifier of the volume.

volume_type

The type of Volume target.

openstack.baremetal.v1.deploy_templates

The DeployTemplate Class

The `DeployTemplate` class inherits from *Resource*.

class openstack.baremetal.v1.deploy_templates.**DeployTemplate**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'deploy_templates'

Plural form of key for resource.

base_path = '/deploy_templates'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_patch = True

Allow patch operation for this resource.

commit_method = 'PATCH'

Method for committing a resource (PUT, PATCH, POST)

commit_jsonpatch = True

Whether commit uses JSON patch format.

name

The name of this resource.

created_at

Timestamp at which the deploy_template was created.

extra

A set of one or more arbitrary metadata key and value pairs.

links

A list of relative links. Includes the self and bookmark links.

steps

A set of physical information of the deploy_template.

updated_at

Timestamp at which the deploy_template was last updated.

id

The UUID of the resource.

openstack.baremetal.v1.conductor

The Conductor Class

The `Conductor` class inherits from *`Resource`*.

class openstack.baremetal.v1.conductor.**Conductor**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'conductors'

Plural form of key for resource.

base_path = '/conductors'

The base part of the URI for this resource.

allow_create = False

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = False

Allow update operation for this resource.

allow_delete = False

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_patch = False

Allow patch operation for this resource.

Baremetal Introspection Resources

openstack.baremetal_introspection.v1.Introspection

The Introspection Class

The `Introspection` class inherits from *Resource*.

class openstack.baremetal_introspection.v1.introspection.**Introspection**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.

- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'introspection'

Plural form of key for resource.

base_path = '/introspection'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = False

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

create_method = 'POST'

Method for creating a resource (POST, PUT)

create_requires_id = True

Whether create requires an ID (determined from method if None).

create_returns_body = False

Does create returns a body (if False requires ID), defaults to has_body

finished_at

Timestamp at which the introspection was finished.

error

The last error message (if any).

id

The UUID of the introspection (matches the node UUID).

is_finished

Whether introspection is finished.

links

A list of relative links, including the self and bookmark links.

started_at

Timestamp at which the introspection was started.

state

The current introspection state.

abort(*session*)

Abort introspection.

Parameters
session (Adapter) The session to use for making this request.

get_data(*session*, *processed=True*)

Get introspection data.

Note that the introspection data format is not stable and can vary from environment to environment.

Parameters

- **session** (Adapter) The session to use for making this request.
- **processed** (*bool*) Whether to fetch the final processed data (the default) or the raw unprocessed data as received from the ramdisk.

Returns
introspection data from the most recent successful run.

Return type
dict

wait(*session*, *timeout=None*, *ignore_error=False*)

Wait for the node to reach the expected state.

Parameters

- **session** (Adapter) The session to use for making this request.
- **timeout** How much (in seconds) to wait for the introspection. The value of None (the default) means no client-side timeout.
- **ignore_error** If True, this call will raise an exception if the introspection reaches the error state. Otherwise the error state is considered successful and the call returns.

Returns
This *Introspection* instance.

Raises
ResourceFailure if introspection fails and ignore_error is False.

Raises
ResourceTimeout on timeout.

openstack.baremetal_introspection.v1.introspection_rule

The IntrospectionRule Class

The IntrospectionRule class inherits from *Resource*.

class openstack.baremetal_introspection.v1.introspection_rule.**IntrospectionRule**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'rules'

Plural form of key for resource.

base_path = '/rules'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = False

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

create_method = 'POST'

Method for creating a resource (POST, PUT)

create_requires_id = True

Whether create requires an ID (determined from method if None).

id

The UUID of the resource.

conditions

List of a logic statementd or operations in rules

actions

List of operations that will be performed if conditions of this rule are fulfilled.

description

Rule human-readable description

scope

Scope of an introspection rule

links

A list of relative links, including the self and bookmark links.

Block Storage Resources

Block Storage v2 Resources

openstack.block_storage.v2.backup

The Backup Class

The Backup class inherits from *Resource*.

class openstack.block_storage.v2.backup.**Backup**(*_synchronized=False*, *connection=None*, ***attrs*)

Volume Backup

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of self._connection in Resource code should protect itself with a check for None.

resource_key = 'backup'

Singular form of key for resource.

resources_key = 'backups'

Plural form of key for resource.

base_path = '/backups'

The base part of the URI for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_create = True

Allow create operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

availability_zone

Properties backup availability zone

container

The container backup in

created_at

The date and time when the resource was created.

data_timestamp

data timestamp The time when the data on the volume was first saved. If it is a backup from volume, it will be the same as created_at for a backup. If it is a backup from a snapshot, it will be the same as created_at for the snapshot.

description

backup description

fail_reason

Backup fail reason

force

Force backup

has_dependent_backups

has_dependent_backups If this value is true, there are other backups depending on this backup.

is_incremental

Indicates whether the backup mode is incremental. If this value is true, the backup mode is incremental. If this value is false, the backup mode is full.

links

A list of links associated with this volume. *Type: list*

name

backup name

object_count

backup object count

size

The size of the volume, in gibibytes (GiB).

snapshot_id

The UUID of the source volume snapshot.

status

backup status values: creating, available, deleting, error, restoring, error_restoring

updated_at

The date and time when the resource was updated.

volume_id

The UUID of the volume.

volume_name

The name of the volume.

create(*session*, *prepend_key=True*, *base_path=None*, ***params*)

Create a remote resource based on this instance.

Parameters

- **session** (Adapter) The session to use for making this request.

- **prepend_key** A boolean indicating whether the resource_key should be prepended in a resource creation request. Default to True.
- **base_path** (*str*) Base part of the URI for creating resources, if different from *base_path*.
- **params** (*dict*) Additional params to pass.

Returns
This `Resource` instance.

Raises
MethodNotSupported if `Resource.allow_create` is not set to `True`.

restore(*session*, *volume_id=None*, *name=None*)

Restore current backup to volume

Parameters

- **session** openstack session
- **volume_id** The ID of the volume to restore the backup to.
- **name** The name for new volume creation to restore.

Returns
Updated backup instance

force_delete(*session*)

Force backup deletion

reset(*session*, *status*)

Reset the status of the backup

openstack.block_storage.v2.capabilities

The Capabilities Class

The `Capabilities` class inherits from *Resource*.

class openstack.block_storage.v2.capabilities.**Capabilities**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

base_path = '/capabilities'

The base part of the URI for this resource.

allow_fetch = True

Allow get operation for this resource.

description

Properties The capabilities description

display_name

The name of volume backend capabilities.

driver_version

The driver version.

namespace

The storage namespace, such as OS::Storage::Capabilities::foo.

pool_name

The name of the storage pool.

properties

The backend volume capabilites list, which consists of cinder standard capabilities and vendor unique properties.

replication_targets

A list of volume backends used to replicate volumes on this backend.

storage_protocol

The storage backend for the backend volume.

vendor_name

The name of the vendor.

visibility

The volume type access.

volume_backend_name

The name of the back-end volume.

openstack.block_storage.v2.limits

The AbsoluteLimit Class

The `AbsoluteLimit` class inherits from *Resource*.

class openstack.block_storage.v2.limits.**AbsoluteLimit**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

max_total_backup_gigabytes

Properties The maximum total amount of backups, in gibibytes (GiB).

max_total_backups

The maximum number of backups.

max_total_snapshots

The maximum number of snapshots.

max_total_volume_gigabytes

The maximum total amount of volumes, in gibibytes (GiB).

max_total_volumes

The maximum number of volumes.

total_backup_gigabytes_used

The total number of backups gibibytes (GiB) used.

total_backups_used

The total number of backups used.

total_gigabytes_used

The total number of gibibytes (GiB) used.

total_snapshots_used

The total number of snapshots used.

total_volumes_used

The total number of volumes used.

The Limits Class

The `Limit` class inherits from *Resource*.

class openstack.block_storage.v2.limits.**Limits**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'limits'

Singular form of key for resource.

base_path = '/limits'

The base part of the URI for this resource.

allow_fetch = True

Allow get operation for this resource.

absolute

Properties An absolute limits object.

rate

Rate-limit volume copy bandwidth, used to mitigate slow down of data access from the instances.

The RateLimit Class

The `RateLimit` class inherits from *Resource*.

class openstack.block_storage.v2.limits.**RateLimit**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

next_available

Properties Rate limits next availabe time.

remaining

Integer for rate limits remaining.

unit

Unit of measurement for the value parameter.

value

Integer number of requests which can be made.

verb

An HTTP verb (POST, PUT, etc.).

The RateLimits Class

The `RateLimits` class inherits from *Resource*.

class openstack.block_storage.v2.limits.**RateLimits**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

limits

Properties A list of the specific limits that apply to the `regex` and `uri`.

regex

A regex representing which routes this rate limit applies to.

uri

A URI representing which routes this rate limit applies to.

openstack.block_storage.v2.quota_set

The QuotaSet Class

The `QuotaSet` class inherits from *Resource*.

class openstack.block_storage.v2.quota_set.**QuotaSet**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

backup_gigabytes

Properties The size (GB) of backups that are allowed for each project.

backups

The number of backups that are allowed for each project.

gigabytes

The size (GB) of volumes and snapshots that are allowed for each project.

groups

The number of groups that are allowed for each project.

per_volume_gigabytes

The size (GB) of volumes in request that are allowed for each volume.

snapshots

The number of snapshots that are allowed for each project.

volumes

The number of volumes that are allowed for each project.

openstack.block_storage.v2.snapshot

The Snapshot Class

The `Snapshot` class inherits from *Resource*.

class openstack.block_storage.v2.snapshot.**Snapshot**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'snapshot'

Singular form of key for resource.

resources_key = 'snapshots'

Plural form of key for resource.

base_path = '/snapshots'

The base part of the URI for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_create = True

Allow create operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_list = True

Allow list operation for this resource.

created_at

The timestamp of this snapshot creation.

description

Description of snapshot. Default is None.

is_forced

Indicate whether to create snapshot, even if the volume is attached. Default is `False`. *Type: bool*

size

The size of the volume, in GBs.

status

The current status of this snapshot. Potential values are creating, available, deleting, error, and error_deleting.

updated_at

The date and time when the resource was updated.

volume_id

The ID of the volume this snapshot was taken of.

reset(*session*, *status*)

Reset the status of the snapshot.

The SnapshotDetail Class

The `SnapshotDetail` class inherits from *Snapshot*.

`openstack.block_storage.v2.snapshot.`**SnapshotDetail**

alias of *Snapshot*

openstack.block_storage.v2.stats

The Pools Class

The `Pools` class inherits from *Resource*.

class `openstack.block_storage.v2.stats.`**Pools**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (`openstack.connection.Connection`) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = ''

Singular form of key for resource.

resources_key = 'pools'

Plural form of key for resource.

base_path = '/scheduler-stats/get_pools?detail=True'

The base part of the URI for this resource.

allow_fetch = False

Allow get operation for this resource.

allow_create = False

Allow create operation for this resource.

allow_delete = False

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

name

The Cinder name for the pool

capabilities

returns a dict with information about the pool

openstack.block_storage.v2.type

The Type Class

The Type class inherits from *Resource*.

class openstack.block_storage.v2.type.**Type**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'volume_type'

Singular form of key for resource.

resources_key = 'volume_types'

Plural form of key for resource.

base_path = '/types'

The base part of the URI for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_create = True

Allow create operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

extra_specs

A dict of extra specifications. capabilities is a usual key.

is_public

a private volume-type. *Type: bool*

get_private_access(*session*)

List projects with private access to the volume type.

Parameters
session The session to use for making this request.

Returns
The volume type access response.

add_private_access(*session*, *project_id*)

Add project access from the volume type.

Parameters

- **session** The session to use for making this request.
- **project_id** The project to add access for.

remove_private_access(*session*, *project_id*)

Remove project access from the volume type.

Parameters

- **session** The session to use for making this request.
- **project_id** The project to remove access for.

openstack.block_storage.v2.volume

The Volume Class

The `Volume` class inherits from *Resource*.

class openstack.block_storage.v2.volume.**Volume**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'volume'

Singular form of key for resource.

resources_key = 'volumes'

Plural form of key for resource.

base_path = '/volumes'

The base part of the URI for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_create = True

Allow create operation for this resource.

`allow_delete = True`

Allow delete operation for this resource.

`allow_commit = True`

Allow update operation for this resource.

`allow_list = True`

Allow list operation for this resource.

`attachments`

TODO(briancurtin): This is currently undocumented in the API.

`availability_zone`

The availability zone.

`consistency_group_id`

ID of the consistency group.

`created_at`

The timestamp of this volume creation.

`updated_at`

The date and time when the resource was updated.

`description`

The volume description.

`extended_replication_status`

Extended replication status on this volume.

`host`

The volumes current back-end.

`image_id`

The ID of the image from which you want to create the volume. Required to create a bootable volume.

`is_bootable`

Enables or disables the bootable attribute. You can boot an instance from a bootable volume. *Type: bool*

`is_encrypted`

`True` if this volume is encrypted, `False` if not. *Type: bool*

`migration_id`

The volume ID that this volumes name on the back-end is based on.

`migration_status`

The status of this volumes migration (None means that a migration is not currently in progress).

`project_id`

The project ID associated with current back-end.

`replication_driver_data`

Data set by the replication driver

replication_status

Status of replication on this volume.

scheduler_hints

Scheduler hints for the volume

size

The size of the volume, in GBs. *Type: int*

snapshot_id

To create a volume from an existing snapshot, specify the ID of the existing volume snapshot. If specified, the volume is created in same availability zone and with same size of the snapshot.

source_volume_id

To create a volume from an existing volume, specify the ID of the existing volume. If specified, the volume is created with same size of the source volume.

status

One of the following values: creating, available, attaching, in-use deleting, error, error_deleting, backing-up, restoring-backup, error_restoring. For details on these statuses, see the Block Storage API documentation.

user_id

The user ID associated with the volume

volume_image_metadata

One or more metadata key and value pairs about image

volume_type

The name of the associated volume type.

extend(*session*, *size*)

Extend a volume size.

set_bootable_status(*session*, *bootable=True*)

Set volume bootable status flag

set_readonly(*session*, *readonly*)

Set volume readonly flag

set_image_metadata(*session*, *metadata*)

Sets image metadata key-value pairs on the volume

delete_image_metadata(*session*)

Remove all image metadata from the volume

delete_image_metadata_item(*session*, *key*)

Remove a single image metadata from the volume

reset_status(*session*, *status=None*, *attach_status=None*, *migration_status=None*)

Reset volume statuses (admin operation)

attach(*session*, *mountpoint*, *instance*)

Attach volume to server

detach(*session*, *attachment*, *force=False*)

Detach volume from server

unmanage(*session*)

Unmanage volume

retype(*session*, *new_type*, *migration_policy=None*)

Change volume type

migrate(*session*, *host=None*, *force_host_copy=False*, *lock_volume=False*)

Migrate volume

complete_migration(*session*, *new_volume_id*, *error=False*)

Complete volume migration

force_delete(*session*)

Force volume deletion

Block Storage v3 Resources

openstack.block_storage.v3.attachment

The Volume Attachment Class

The `Volume Attachment` class inherits from *Resource*.

class openstack.block_storage.v3.attachment.**Attachment**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

create(*session*, *prepend_key=True*, *base_path=None*, ***, *resource_request_key=None*, *resource_response_key=None*, *microversion=None*, ***params*)

Create a remote resource based on this instance.

Parameters

- **session** (`Adapter`) The session to use for making this request.
- **prepend_key** A boolean indicating whether the resource_key should be prepended in a resource creation request. Default to True.
- **base_path** (*str*) Base part of the URI for creating resources, if different from *base_path*.
- **resource_request_key** (*str*) Overrides the usage of self.resource_key when prepending a key to the request body. Ignored if *prepend_key* is false.

- **resource_response_key** (*str*) Overrides the usage of self.resource_key when processing response bodies. Ignored if *prepend_key* is false.
- **microversion** (*str*) API version to override the negotiated one.
- **params** (*dict*) Additional params to pass.

Returns
This `Resource` instance.

Raises
MethodNotSupported if `Resource.allow_create` is not set to `True`.

complete(*session*, *, *microversion=None*)

Mark the attachment as completed.

update(*[E,]**F*) → None. Update D from dict/iterable E and F.

If E is present and has a .keys() method, then does: for k in E: D[k] = E[k] If E is present and lacks a .keys() method, then does: for k, v in E: D[k] = v In either case, this is followed by: for k in F: D[k] = F[k]

openstack.block_storage.v3.availability_zone

The AvailabilityZone Class

The `AvailabilityZone` class inherits from *Resource*.

class openstack.block_storage.v3.availability_zone.**AvailabilityZone**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (*openstack.connection.Connection*) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = ''

Singular form of key for resource.

resources_key = 'availabilityZoneInfo'

Plural form of key for resource.

base_path = '/os-availability-zone'

The base part of the URI for this resource.

allow_list = True

Allow list operation for this resource.

name

Properties Name of availability zone

state

State of availability zone, available is usual key

openstack.block_storage.v3.backup

The Backup Class

The Backup class inherits from *Resource*.

class openstack.block_storage.v3.backup.**Backup**(*_synchronized=False*, *connection=None*, ***attrs*)

Volume Backup

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'backup'

Singular form of key for resource.

resources_key = 'backups'

Plural form of key for resource.

base_path = '/backups'

The base part of the URI for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_create = True

Allow create operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

availability_zone

Properties backup availability zone

container

The container backup in

created_at

The date and time when the resource was created.

data_timestamp

data timestamp The time when the data on the volume was first saved. If it is a backup from volume, it will be the same as created_at for a backup. If it is a backup from a snapshot, it will be the same as created_at for the snapshot.

description

backup description

encryption_key_id

The UUID of the encryption key. Only included for encrypted volumes.

fail_reason

Backup fail reason

force

Force backup

has_dependent_backups

has_dependent_backups If this value is true, there are other backups depending on this backup.

is_incremental

Indicates whether the backup mode is incremental. If this value is true, the backup mode is incremental. If this value is false, the backup mode is full.

links

A list of links associated with this volume. *Type: list*

metadata

The backup metadata. New in version 3.43

name

backup name

object_count

backup object count

project_id

The UUID of the owning project. New in version 3.18

size

The size of the volume, in gibibytes (GiB).

snapshot_id

The UUID of the source volume snapshot.

status

backup status values: creating, available, deleting, error, restoring, error_restoring

updated_at

The date and time when the resource was updated.

user_id

The UUID of the project owner. New in 3.56

volume_id

The UUID of the volume.

volume_name

The name of the volume.

create(*session*, *prepend_key=True*, *base_path=None*, ***params*)

Create a remote resource based on this instance.

Parameters

- **session** (Adapter) The session to use for making this request.
- **prepend_key** A boolean indicating whether the resource_key should be prepended in a resource creation request. Default to True.
- **base_path** (*str*) Base part of the URI for creating resources, if different from *base_path*.
- **params** (*dict*) Additional params to pass.

Returns

This Resource instance.

Raises

MethodNotSupported if Resource.allow_create is not set to True.

restore(*session*, *volume_id=None*, *name=None*)

Restore current backup to volume

Parameters

- **session** openstack session
- **volume_id** The ID of the volume to restore the backup to.
- **name** The name for new volume creation to restore.

Returns

Updated backup instance

force_delete(*session*)

Force backup deletion

reset(*session*, *status*)

Reset the status of the backup

openstack.block_storage.v3.block_storage_summary

The Block Storage Summary Class

The Block Storage Summary class inherits from *Resource*.

class openstack.block_storage.v3.block_storage_summary.**BlockStorageSummary**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

base_path = '/volumes/summary'

The base part of the URI for this resource.

allow_fetch = True

Allow get operation for this resource.

total_size

Total size of all the volumes

total_count

Total count of all the volumes

metadata

Metadata of all the volumes

openstack.block_storage.v3.capabilities

The Capabilities Class

The `Capabilities` class inherits from *Resource*.

class openstack.block_storage.v3.capabilities.**Capabilities**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

base_path = '/capabilities'

The base part of the URI for this resource.

allow_fetch = True

Allow get operation for this resource.

description

Properties The capabilities description

display_name

The name of volume backend capabilities.

driver_version

The driver version.

namespace

The storage namespace, such as OS::Storage::Capabilities::foo.

pool_name

The name of the storage pool.

properties

The backend volume capabilites list, which consists of cinder standard capabilities and vendor unique properties.

replication_targets

A list of volume backends used to replicate volumes on this backend.

storage_protocol

The storage backend for the backend volume.

vendor_name

The name of the vendor.

visibility

The volume type access.

volume_backend_name

The name of the back-end volume.

openstack.block_storage.v3.extension

The Extension Class

The `Extension` class inherits from *Resource*.

class openstack.block_storage.v3.extension.**Extension**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'extensions'

Plural form of key for resource.

base_path = '/extensions'

The base part of the URI for this resource.

allow_list = True

Allow list operation for this resource.

alias

Properties The alias for the extension.

description

The extension description.

links

Links pertaining to this extension.

name

The name of this extension.

updated_at

The date and time when the resource was updated. The date and time stamp format is ISO 8601.

openstack.block_storage.v3.group

The Group Class

The Group class inherits from *Resource*.

class openstack.block_storage.v3.group.**Group**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'group'

Singular form of key for resource.

resources_key = 'groups'

Plural form of key for resource.

base_path = '/groups'

The base part of the URI for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_create = True

Allow create operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_list = True

Allow list operation for this resource.

delete(*session*, **args*, *delete_volumes=False*, ***kwargs*)

Delete a group.

reset(*session*, *status*)

Resets the status for a group.

classmethod create_from_source(*session*, *group_snapshot_id*, *source_group_id*, *name=None*, *description=None*)

Creates a new group from source.

openstack.block_storage.v3.group_snapshot

The GroupSnapshot Class

The GroupSnapshot class inherits from *Resource*.

class openstack.block_storage.v3.group_snapshot.**GroupSnapshot**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of self._connection in Resource code should protect itself with a check for None.

resource_key = 'group_snapshot'

Singular form of key for resource.

resources_key = 'group_snapshots'

Plural form of key for resource.

base_path = '/group_snapshots'

The base part of the URI for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_create = True

Allow create operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_commit = False

Allow update operation for this resource.

allow_list = True

Allow list operation for this resource.

created_at

Properties The date and time when the resource was created.

description

The group snapshot description.

group_id

The UUID of the source group.

group_type_id

The group type ID.

id

The ID of the group snapshot.

name

The group snapshot name.

project_id

The UUID of the volume group snapshot project.

status

The status of the generic group snapshot.

reset_state(*session*, *state*)

Resets the status for a group snapshot.

openstack.block_storage.v3.group_type

The GroupType Class

The GroupType class inherits from *Resource*.

class openstack.block_storage.v3.group_type.**GroupType**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of self._connection in Resource code should protect itself with a check for None.

resource_key = 'group_type'

Singular form of key for resource.

`resources_key = 'group_types'`

Plural form of key for resource.

`base_path = '/group_types'`

The base part of the URI for this resource.

`allow_fetch = True`

Allow get operation for this resource.

`allow_create = True`

Allow create operation for this resource.

`allow_delete = True`

Allow delete operation for this resource.

`allow_commit = True`

Allow update operation for this resource.

`allow_list = True`

Allow list operation for this resource.

`description`

Properties The group type description.

`group_specs`

Contains the specifications for a group type.

`is_public`

Whether the group type is publicly visible.

`fetch_group_specs`(*session*)

Fetch group_specs of the group type.

These are returned by default if the user has suitable permissions (i.e. youre an admin) but by default you also need the same permissions to access this API. That means this function is kind of useless. However, that is how the API was designed and it is theoretically possible that people will have modified their policy to allow this but not the other so we provide this anyway.

Parameters

`session` The session to use for making this request.

Returns

An updated version of this object.

`create_group_specs`(*session*, *specs*)

Creates group specs for the group type.

This will override whatever specs are already present on the group type.

Parameters

- `session` The session to use for making this request.
- `specs` A dict of group specs to set on the group type.

Returns

An updated version of this object.

get_group_specs_property(*session*, *prop*)

Retrieve a group spec property of the group type.

Parameters

- **session** The session to use for making this request.
- **prop** The name of the group spec property to update.

Returns

The value of the group spec property.

update_group_specs_property(*session*, *prop*, *val*)

Update a group spec property of the group type.

Parameters

- **session** The session to use for making this request.
- **prop** The name of the group spec property to update.
- **val** The value to set for the group spec property.

Returns

The updated value of the group spec property.

delete_group_specs_property(*session*, *prop*)

Delete a group spec property from the group type.

Parameters

- **session** The session to use for making this request.
- **prop** The name of the group spec property to delete.

Returns

None

openstack.block_storage.v3.limits

The AbsoluteLimit Class

The `AbsoluteLimit` class inherits from *Resource*.

class openstack.block_storage.v3.limits.**AbsoluteLimit**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

max_total_backup_gigabytes

The maximum total amount of backups, in gibibytes (GiB).

max_total_backups

The maximum number of backups.

max_total_snapshots

The maximum number of snapshots.

max_total_volume_gigabytes

The maximum total amount of volumes, in gibibytes (GiB).

max_total_volumes

The maximum number of volumes.

total_backup_gigabytes_used

The total number of backups gibibytes (GiB) used.

total_backups_used

The total number of backups used.

total_gigabytes_used

The total number of gibibytes (GiB) used.

total_snapshots_used

The total number of snapshots used.

total_volumes_used

The total number of volumes used.

The Limits Class

The `Limits` class inherits from *Resource*.

class openstack.block_storage.v3.limits.**Limits**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'limits'

Singular form of key for resource.

base_path = '/limits'

The base part of the URI for this resource.

allow_fetch = True

Allow get operation for this resource.

absolute

An absolute limits object.

rate

Rate-limit volume copy bandwidth, used to mitigate slow down of data access from the instances.

The RateLimit Class

The `RateLimit` class inherits from *Resource*.

class openstack.block_storage.v3.limits.**RateLimit**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

next_available

Rate limits next availabe time.

remaining

Integer for rate limits remaining.

unit

Unit of measurement for the value parameter.

value

Integer number of requests which can be made.

verb

An HTTP verb (POST, PUT, etc.).

The RateLimits Class

The `RateLimits` class inherits from *Resource*.

class openstack.block_storage.v3.limits.**RateLimits**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

limits

A list of the specific limits that apply to the `regex` and `uri`.

regex

A regex representing which routes this rate limit applies to.

uri

A URI representing which routes this rate limit applies to.

openstack.block_storage.v3.quota_set

The QuotaSet Class

The `QuotaSet` class inherits from *Resource*.

class openstack.block_storage.v3.quota_set.**QuotaSet**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

backup_gigabytes

Properties The size (GB) of backups that are allowed for each project.

backups

The number of backups that are allowed for each project.

gigabytes

The size (GB) of volumes and snapshots that are allowed for each project.

groups

The number of groups that are allowed for each project.

per_volume_gigabytes

The size (GB) of volumes in request that are allowed for each volume.

snapshots

The number of snapshots that are allowed for each project.

volumes

The number of volumes that are allowed for each project.

openstack.block_storage.v3.resource_filter

The ResourceFilter Class

The `ResourceFilter` class inherits from *Resource*.

class openstack.block_storage.v3.resource_filter.**ResourceFilter**(*_synchronized=False*, *connection=None*, ***attrs*)

Resource Filter

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'resource_filters'

Plural form of key for resource.

base_path = '/resource_filters'

The base part of the URI for this resource.

allow_list = True

Allow list operation for this resource.

filters

Properties The list of filters that are applicable to the specified resource.

resource

The resource that the filters will be applied to.

openstack.block_storage.v3.service

The Service Class

The `Service` class inherits from *Resource*.

class openstack.block_storage.v3.service.**Service**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'services'

Plural form of key for resource.

base_path = '/os-services'

The base part of the URI for this resource.

allow_list = True

Allow list operation for this resource.

active_backend_id

The ID of active storage backend (cinder-volume services only)

availability_zone

The availability zone of service

backend_state

The state of storage backend (cinder-volume services only)

binary

Binary name of service

cluster

The cluster name (since 3.7)

disabled_reason

Disabled reason of service

host

The name of the host where service runs

name

Service name

replication_status

The volume service replication status (cinder-volume services only)

state

State of service

status

Status of service

updated_at

The date and time when the resource was updated

classmethod find(*session*, *name_or_id*, *ignore_missing=True*, ***params*)

Find a resource by its name or id.

Parameters

- **session** (`Adapter`) The session to use for making this request.
- **name_or_id** This resources identifier, if needed by the request. The default is `None`.
- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **list_base_path** (*str*) base_path to be used when need listing resources.
- **microversion** (*str*) API version to override the negotiated one.
- **params** (*dict*) Any additional parameters to be passed into underlying methods, such as to *existing()* in order to pass on URI parameters.

Returns
The `Resource` object matching the given name or id or None if nothing matches.

Raises
openstack.exceptions.DuplicateResource if more than one resource is found for this request.

Raises
openstack.exceptions.NotFoundException if nothing is found and ignore_missing is `False`.

commit(*session*, *prepend_key=False*, **args*, ***kwargs*)

Commit the state of the instance to the remote resource.

Parameters

- **session** (`Adapter`) The session to use for making this request.
- **prepend_key** A boolean indicating whether the resource_key should be prepended in a resource update request. Default to True.
- **retry_on_conflict** (*bool*) Whether to enable retries on HTTP CONFLICT (409). Value of `None` leaves the *Adapter* defaults.
- **base_path** (*str*) Base part of the URI for modifying resources, if different from *base_path*.
- **microversion** (*str*) API version to override the negotiated one.
- **kwargs** (*dict*) Parameters that will be passed to _prepare_request()

Returns
This `Resource` instance.

Raises
MethodNotSupported if `Resource.allow_commit` is not set to `True`.

enable(*session*)

Enable service.

disable(*session*, ***, *reason=None*)

Disable service.

failover(*session*, ***, *cluster=None*, *backend_id=None*)

Failover a service

Only applies to replicating cinder-volume services.

openstack.block_storage.v3.snapshot

The Snapshot Class

The `Snapshot` class inherits from *Resource*.

class openstack.block_storage.v3.snapshot.**Snapshot**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'snapshot'

Singular form of key for resource.

resources_key = 'snapshots'

Plural form of key for resource.

base_path = '/snapshots'

The base part of the URI for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_create = True

Allow create operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_list = True

Allow list operation for this resource.

consumes_quota

Whether this resource consumes quota or not. Resources that not counted for quota usage are usually temporary internal resources created to perform an operation. This is included from microversion 3.65

created_at

The timestamp of this snapshot creation.

description

Description of snapshot. Default is None.

group_snapshot_id

The ID of the group snapshot. This is included from microversion 3.14

is_forced

Indicate whether to create snapshot, even if the volume is attached. Default is `False`. *Type: bool*

progress

The percentage of completeness the snapshot is currently at.

project_id

The project ID this snapshot is associated with.

size

The size of the volume, in GBs.

status

The current status of this snapshot. Potential values are creating, available, deleting, error, and error_deleting.

updated_at

The date and time when the resource was updated.

user_id

The UUID of the user. This is included from microversion 3.41

volume_id

The ID of the volume this snapshot was taken of.

force_delete(*session*)

Force snapshot deletion.

reset(*session*, *status*)

Reset the status of the snapshot.

set_status(*session*, *status*, *progress=None*)

Update fields related to the status of a snapshot.

classmethod manage(*session*, *volume_id*, *ref*, *name=None*, *description=None*, *metadata=None*)

Manage a snapshot under block storage provisioning.

unmanage(*session*)

Unmanage a snapshot from block storage provisioning.

The SnapshotDetail Class

The `SnapshotDetail` class inherits from *Snapshot*.

`openstack.block_storage.v3.snapshot.`**SnapshotDetail**

alias of *Snapshot*

openstack.block_storage.v3.stats

The Pools Class

The `Pools` class inherits from *Resource*.

class `openstack.block_storage.v3.stats.`**Pools**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.

- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = ''

Singular form of key for resource.

resources_key = 'pools'

Plural form of key for resource.

base_path = '/scheduler-stats/get_pools?detail=True'

The base part of the URI for this resource.

allow_fetch = False

Allow get operation for this resource.

allow_create = False

Allow create operation for this resource.

allow_delete = False

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

name

The Cinder name for the pool

capabilities

returns a dict with information about the pool

openstack.block_storage.v3.transfer

The Volume Transfer Class

The `Volume Transfer` class inherits from *Resource*.

class openstack.block_storage.v3.transfer.**Transfer**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'transfer'

Singular form of key for resource.

resources_key = 'transfers'

Plural form of key for resource.

base_path = '/volume-transfers'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_list = True

Allow list operation for this resource.

id

UUID of the transfer.

created_at

The date and time when the resource was created.

name

Name of the volume to transfer.

volume_id

ID of the volume to transfer.

auth_key

Auth key for the transfer.

links

A list of links associated with this volume. *Type: list*

no_snapshots

Whether to transfer snapshots or not

create(*session*, *prepend_key=True*, *base_path=None*, ***, *resource_request_key=None*, *resource_response_key=None*, *microversion=None*, ***params*)

Create a volume transfer.

Parameters

- **session** (Adapter) The session to use for making this request.
- **prepend_key** A boolean indicating whether the resource_key should be prepended in a resource creation request. Default to True.
- **base_path** (*str*) Base part of the URI for creating resources, if different from *base_path*.
- **resource_request_key** (*str*) Overrides the usage of self.resource_key when prepending a key to the request body. Ignored if *prepend_key* is false.
- **resource_response_key** (*str*) Overrides the usage of self.resource_key when processing response bodies. Ignored if *prepend_key* is false.

- **microversion** (*str*) API version to override the negotiated one.
- **params** (*dict*) Additional params to pass.

Returns
This Resource instance.

Raises
MethodNotSupported if Resource.allow_create is not set to True.

fetch(*session*, *requires_id=True*, *base_path=None*, *error_message=None*, *skip_cache=False*, *, *resource_response_key=None*, *microversion=None*, ***params*)

Get volume transfer.

Parameters

- **session** (Adapter) The session to use for making this request.
- **requires_id** (*boolean*) A boolean indicating whether resource ID should be part of the requested URI.
- **base_path** (*str*) Base part of the URI for fetching resources, if different from *base_path*.
- **error_message** (*str*) An Error message to be returned if requested object does not exist.
- **skip_cache** (*bool*) A boolean indicating whether optional API cache should be skipped for this invocation.
- **resource_response_key** (*str*) Overrides the usage of self.resource_key when processing the response body.
- **microversion** (*str*) API version to override the negotiated one.
- **params** (*dict*) Additional parameters that can be consumed.

Returns
This Resource instance.

Raises
MethodNotSupported if Resource.allow_fetch is not set to True.

Raises
NotFoundException if the resource was not found.

delete(*session*, *error_message=None*, *, *microversion=None*, ***kwargs*)

Delete a volume transfer.

Parameters

- **session** (Adapter) The session to use for making this request.
- **microversion** (*str*) API version to override the negotiated one.
- **kwargs** (*dict*) Parameters that will be passed to _prepare_request()

Returns
This Resource instance.

Raises
MethodNotSupported if Resource.allow_commit is not set to True.

Raises
NotFoundException if the resource was not found.

accept(*session*, ***, *auth_key=None*)
Accept a volume transfer.

Parameters

- **session** The session to use for making this request.
- **auth_key** The authentication key for the volume transfer.

Returns
This *Transfer* instance.

openstack.block_storage.v3.type

The Type Class

The Type class inherits from *Resource*.

class openstack.block_storage.v3.type.**Type**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'volume_type'
Singular form of key for resource.

resources_key = 'volume_types'
Plural form of key for resource.

base_path = '/types'
The base part of the URI for this resource.

allow_fetch = True
Allow get operation for this resource.

allow_create = True
Allow create operation for this resource.

allow_delete = True
Allow delete operation for this resource.

allow_list = True
Allow list operation for this resource.

allow_commit = True
Allow update operation for this resource.

description
Description of the type.

extra_specs
A dict of extra specifications. capabilities is a usual key.

is_public
a private volume-type. *Type: bool*

set_extra_specs(*session*, ***extra_specs*)
Update extra specs.

This call will replace only the extra_specs with the same keys given here. Other keys will not be modified.

Parameters

- **session** The session to use for making this request.
- **extra_specs** (*kwargs*) Key/value extra_specs pairs to be update on this volume type. All keys and values.

Returns
The updated extra specs.

delete_extra_specs(*session*, *keys*)
Delete extra specs.

Note

This method will do a HTTP DELETE request for every key in keys.

Parameters

- **session** The session to use for this request.
- **keys** (*list*) The keys to delete.

Returns
None

get_private_access(*session*)
List projects with private access to the volume type.

Parameters
session The session to use for making this request.

Returns
The volume type access response.

add_private_access(*session*, *project_id*)
Add project access from the volume type.

Parameters

- **session** The session to use for making this request.
- **project_id** The project to add access for.

remove_private_access(*session*, *project_id*)

Remove project access from the volume type.

Parameters

- **session** The session to use for making this request.
- **project_id** The project to remove access for.

The TypeEncryption Class

The TypeEncryption class inherits from *Resource*.

class openstack.block_storage.v3.type.**TypeEncryption**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of self._connection in Resource code should protect itself with a check for None.

resource_key = 'encryption'

Singular form of key for resource.

resources_key = 'encryption'

Plural form of key for resource.

base_path = '/types/%(volume_type_id)s/encryption'

The base part of the URI for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_create = True

Allow create operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = False

Allow list operation for this resource.

allow_commit = True

Allow update operation for this resource.

cipher

The encryption algorithm or mode.

control_location

Notional service where encryption is performed.

created_at

The date and time when the resource was created.

deleted

The resource is deleted or not.

deleted_at

The date and time when the resource was deleted.

encryption_id

A ID representing this type.

key_size

The Size of encryption key.

provider

The class that provides encryption support.

updated_at

The date and time when the resource was updated.

volume_type_id

The ID of the Volume Type.

openstack.block_storage.v3.volume

The Volume Class

The `Volume` class inherits from *Resource*.

class openstack.block_storage.v3.volume.**Volume**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'volume'

Singular form of key for resource.

resources_key = 'volumes'

Plural form of key for resource.

base_path = '/volumes'

The base part of the URI for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_create = True

Allow create operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_list = True

Allow list operation for this resource.

attachments

TODO(briancurtin): This is currently undocumented in the API.

availability_zone

The availability zone.

consistency_group_id

ID of the consistency group.

created_at

The timestamp of this volume creation.

updated_at

The date and time when the resource was updated.

description

The volume description.

extended_replication_status

Extended replication status on this volume.

group_id

The ID of the group that the volume belongs to.

host

The volumes current back-end.

image_id

The ID of the image from which you want to create the volume. Required to create a bootable volume.

is_bootable

Enables or disables the bootable attribute. You can boot an instance from a bootable volume. *Type: bool*

is_encrypted

`True` if this volume is encrypted, `False` if not. *Type: bool*

is_multiattach

Whether volume will be sharable or not.

migration_id

The volume ID that this volumes name on the back-end is based on.

migration_status

The status of this volumes migration (None means that a migration is not currently in progress).

project_id

The project ID associated with current back-end.

replication_driver_data

Data set by the replication driver

provider_id

The provider ID for the volume.

replication_status

Status of replication on this volume.

scheduler_hints

Scheduler hints for the volume

size

The size of the volume, in GBs. *Type: int*

snapshot_id

To create a volume from an existing snapshot, specify the ID of the existing volume snapshot. If specified, the volume is created in same availability zone and with same size of the snapshot.

source_volume_id

To create a volume from an existing volume, specify the ID of the existing volume. If specified, the volume is created with same size of the source volume.

status

One of the following values: creating, available, attaching, in-use deleting, error, error_deleting, backing-up, restoring-backup, error_restoring. For details on these statuses, see the Block Storage API documentation.

user_id

The user ID associated with the volume

volume_image_metadata

One or more metadata key and value pairs about image

volume_type

The name of the associated volume type.

extend(*session*, *size*)

Extend a volume size.

set_bootable_status(*session*, *bootable=True*)

Set volume bootable status flag

set_readonly(*session*, *readonly*)

Set volume readonly flag

set_image_metadata(*session*, *metadata*)

Sets image metadata key-value pairs on the volume

delete_image_metadata(*session*)

Remove all image metadata from the volume

delete_image_metadata_item(*session*, *key*)

Remove a single image metadata from the volume

reset_status(*session*, *status=None*, *attach_status=None*, *migration_status=None*)

Reset volume statuses (admin operation)

revert_to_snapshot(*session*, *snapshot_id*)

Revert volume to its snapshot

attach(*session*, *mountpoint*, *instance=None*, *host_name=None*)

Attach volume to server

detach(*session*, *attachment*, *force=False*, *connector=None*)

Detach volume from server

classmethod manage(*session*, *host*, *ref* , *name=None*, *description=None*, *volume_type=None*, *availability_zone=None*, *metadata=None*, *bootable=False*, *cluster=None*)

Manage an existing volume.

unmanage(*session*)

Unmanage volume

retype(*session*, *new_type*, *migration_policy=None*)

Change volume type

migrate(*session*, *host=None*, *force_host_copy=False*, *lock_volume=False*, *cluster=None*)

Migrate volume

complete_migration(*session*, *new_volume_id*, *error=False*)

Complete volume migration

force_delete(*session*)

Force volume deletion

upload_to_image(*session*, *image_name*, *force=False*, *disk_format=None*, *container_format=None*, *visibility=None*, *protected=None*)

Upload the volume to image service

reserve(*session*)

Reserve volume

unreserve(*session*)

Unreserve volume

begin_detaching(*session*)

Update volume status to detaching

abort_detaching(*session*)

Roll back volume status to in-use

init_attachment(*session*, *connector*)

Initialize volume attachment

terminate_attachment(*session*, *connector*)

Terminate volume attachment

Cluster Resources

openstack.clustering.v1.build_info

The BuildInfo Class

The `BuildInfo` class inherits from *Resource*.

class openstack.clustering.v1.build_info.**BuildInfo**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

base_path = '/build-info'

The base part of the URI for this resource.

resource_key = 'build_info'

Singular form of key for resource.

allow_fetch = True

Allow get operation for this resource.

api

String representation of the API build version

engine

String representation of the engine build version

openstack.clustering.v1.profile_type

The ProfileType Class

The `ProfileType` class inherits from *Resource*.

class openstack.clustering.v1.profile_type.**ProfileType**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.

- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'profile_type'

Singular form of key for resource.

resources_key = 'profile_types'

Plural form of key for resource.

base_path = '/profile-types'

The base part of the URI for this resource.

allow_list = True

Allow list operation for this resource.

allow_fetch = True

Allow get operation for this resource.

name

Name of the profile type.

schema

The schema of the profile type.

support_status

The support status of the profile type

openstack.clustering.v1.profile

The Profile Class

The `Profile` class inherits from *Resource*.

class openstack.clustering.v1.profile.**Profile**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'profile'

Singular form of key for resource.

resources_key = 'profiles'

Plural form of key for resource.

base_path = '/profiles'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

commit_method = 'PATCH'

Method for committing a resource (PUT, PATCH, POST)

name

The name of the profile

type

The type of the profile.

project_id

The ID of the project this profile belongs to.

domain_id

The domain ID of the profile.

user_id

The ID of the user who created this profile.

spec

The spec of the profile.

metadata

A collection of key-value pairs that are attached to the profile.

created_at

Timestamp of when the profile was created.

updated_at

Timestamp of when the profile was last updated.

openstack.clustering.v1.policy_type

The PolicyType Class

The `PolicyType` class inherits from *`Resource`*.

class openstack.clustering.v1.policy_type.**PolicyType**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'policy_type'

Singular form of key for resource.

resources_key = 'policy_types'

Plural form of key for resource.

base_path = '/policy-types'

The base part of the URI for this resource.

allow_list = True

Allow list operation for this resource.

allow_fetch = True

Allow get operation for this resource.

name

Name of policy type.

schema

The schema of the policy type.

support_status

The support status of the policy type

openstack.clustering.v1.policy

The Policy Class

The `Policy` class inherits from *Resource*.

class openstack.clustering.v1.policy.**Policy**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'policy'

Singular form of key for resource.

resources_key = 'policies'

Plural form of key for resource.

base_path = '/policies'

The base part of the URI for this resource.

allow_list = True

Allow list operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_create = True

Allow create operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_commit = True

Allow update operation for this resource.

commit_method = 'PATCH'

Method for committing a resource (PUT, PATCH, POST)

name

The name of the policy.

type

The type name of the policy.

project_id

The ID of the project this policy belongs to.

user_id

The ID of the user who created this policy.

created_at

The timestamp when the policy is created.

updated_at

The timestamp when the policy was last updated.

spec

The specification of the policy.

data

A dictionary containing runtime data of the policy.

openstack.clustering.v1.Cluster

The Cluster Class

The `Cluster` class inherits from *`Resource`*.

class openstack.clustering.v1.cluster.**Cluster**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'cluster'

Singular form of key for resource.

resources_key = 'clusters'

Plural form of key for resource.

base_path = '/clusters'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

commit_method = 'PATCH'

Method for committing a resource (PUT, PATCH, POST)

name

The name of the cluster.

profile_id

The ID of the profile used by this cluster.

user_id

The ID of the user who created this cluster, thus the owner of it.

project_id

The ID of the project this cluster belongs to.

domain_id

The domain ID of the cluster owner.

init_at

Timestamp of when the cluster was initialized. *Type: datetime object parsed from ISO 8601 formatted string*

created_at

Timestamp of when the cluster was created. *Type: datetime object parsed from ISO 8601 formatted string*

updated_at

Timestamp of when the cluster was last updated. *Type: datetime object parsed from ISO 8601 formatted string*

min_size

Lower bound (inclusive) for the size of the cluster.

max_size

Upper bound (inclusive) for the size of the cluster. A value of -1 indicates that there is no upper limit of cluster size.

desired_capacity

Desired capacity for the cluster. A cluster would be created at the scale specified by this value.

timeout

Default timeout (in seconds) for cluster operations.

status

A string representation of the cluster status.

status_reason

A string describing the reason why the cluster in current status.

config

A dictionary configuration for cluster.

metadata

A collection of key-value pairs that are attached to the cluster.

data

A dictionary with some runtime data associated with the cluster.

node_ids

A list IDs of nodes that are members of the cluster.

profile_name

Name of the profile used by the cluster.

is_profile_only

Specify whether the cluster update should only pertain to the profile.

dependents

A dictionary with dependency information of the cluster

op(*session*, *operation*, ***params*)

Perform an operation on the cluster.

Parameters

- **session** A session object used for sending request.
- **operation** A string representing the operation to be performed.
- **params** (*dict*) An optional dict providing the parameters for the operation.

Returns
A dictionary containing the action ID.

force_delete(*session*)
Force delete a cluster.

openstack.clustering.v1.Node

The Node Class

The Node class inherits from *Resource*.

class openstack.clustering.v1.node.**Node**(*_synchronized=False*, *connection=None*, ***attrs*)
The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of self._connection in Resource code should protect itself with a check for None.

resource_key = 'node'
Singular form of key for resource.

resources_key = 'nodes'
Plural form of key for resource.

base_path = '/nodes'
The base part of the URI for this resource.

allow_create = True
Allow create operation for this resource.

allow_fetch = True
Allow get operation for this resource.

allow_commit = True
Allow update operation for this resource.

allow_delete = True
Allow delete operation for this resource.

allow_list = True
Allow list operation for this resource.

commit_method = 'PATCH'

Method for committing a resource (PUT, PATCH, POST)

name

The name of the node.

physical_id

The ID of the physical object that backs the node.

cluster_id

The ID of the cluster in which this node is a member. A node is an orphan node if this field is empty.

profile_id

The ID of the profile used by this node.

domain_id

The domain ID of the node.

user_id

The ID of the user who created this node.

project_id

The ID of the project this node belongs to.

profile_name

The name of the profile used by this node.

index

An integer that is unique inside the owning cluster. A value of -1 means this node is an orphan node.

role

A string indicating the role the node plays in a cluster.

init_at

The timestamp of the node objects initialization. *Type: datetime object parsed from ISO 8601 formatted string*

created_at

The timestamp of the nodes creation, i.e. the physical object represented by this node is also created. *Type: datetime object parsed from ISO 8601 formatted string*

updated_at

The timestamp the node was last updated. *Type: datetime object parsed from ISO 8601 formatted string*

status

A string indicating the nodes status.

status_reason

A string describing why the node entered its current status.

metadata

A map containing key-value pairs attached to the node.

data

A map containing some runtime data for this node.

details

A map containing the details of the physical object this node represents

dependents

A map containing the dependency of nodes

tainted

Whether the node is tainted. *Type: bool*

check(*session*, ***params*)

An action procedure for the node to check its health status.

Parameters
session A session object used for sending request.

Returns
A dictionary containing the action ID.

recover(*session*, ***params*)

An action procedure for the node to recover.

Parameters
session A session object used for sending request.

Returns
A dictionary containing the action ID.

op(*session*, *operation*, ***params*)

Perform an operation on the specified node.

Parameters

- **session** A session object used for sending request.
- **operation** A string representing the operation to be performed.
- **params** (*dict*) An optional dict providing the parameters for the operation.

Returns
A dictionary containing the action ID.

adopt(*session*, *preview=False*, ***params*)

Adopt a node for management.

Parameters

- **session** A session object used for sending request.
- **preview** A boolean indicating whether the adoption is a preview. A preview does not create the node object.
- **params** (*dict*) A dict providing the details of a node to be adopted.

force_delete(*session*)

Force delete a node.

openstack.clustering.v1.cluster_policy

The ClusterPolicy Class

The `ClusterPolicy` class inherits from *Resource*.

class openstack.clustering.v1.cluster_policy.**ClusterPolicy**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'cluster_policy'

Singular form of key for resource.

resources_key = 'cluster_policies'

Plural form of key for resource.

base_path = '/clusters/%(cluster_id)s/policies'

The base part of the URI for this resource.

allow_list = True

Allow list operation for this resource.

allow_fetch = True

Allow get operation for this resource.

policy_id

ID of the policy object.

policy_name

Name of the policy object.

cluster_id

ID of the cluster object.

cluster_name

Name of the cluster object.

policy_type

Type string of the policy.

is_enabled

Whether the policy is enabled on the cluster. *Type: bool*

data

Data associated with the cluster-policy binding.

openstack.clustering.v1.receiver

The Receiver Class

The `Receiver` class inherits from *Resource*.

class openstack.clustering.v1.receiver.**Receiver**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'receiver'

Singular form of key for resource.

resources_key = 'receivers'

Plural form of key for resource.

base_path = '/receivers'

The base part of the URI for this resource.

allow_list = True

Allow list operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_create = True

Allow create operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

commit_method = 'PATCH'

Method for committing a resource (PUT, PATCH, POST)

name

The name of the receiver.

type

The type of the receiver.

user_id

The ID of the user who created the receiver, thus the owner of it.

project_id

The ID of the project this receiver belongs to.

domain_id

The domain ID of the receiver.

cluster_id

The ID of the targeted cluster.

action

The name of the targeted action.

created_at

Timestamp of when the receiver was created.

updated_at

Timestamp of when the receiver was last updated.

actor

The credential of the impersonated user.

params

A dictionary containing key-value pairs that are provided to the targeted action.

channel

The information about the channel through which you can trigger the receiver hence the associated action.

openstack.clustering.v1.action

The Action Class

The `Action` class inherits from *Resource*.

`class openstack.clustering.v1.action.`**Action**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (`openstack.connection.Connection`) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'action'

Singular form of key for resource.

resources_key = 'actions'

Plural form of key for resource.

base_path = '/actions'

The base part of the URI for this resource.

allow_list = True

Allow list operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

commit_method = 'PATCH'

Method for committing a resource (PUT, PATCH, POST)

name

Name of the action.

target_id

ID of the target object, which can be a cluster or a node.

action

Built-in type name of action.

cause

A string representation of the reason why the action was created.

owner_id

The owning engine that is currently running the action.

user_id

The ID of the user who created this action.

project_id

The ID of the project this profile belongs to.

domain_id

The domain ID of the action.

interval

Interval in seconds between two consecutive executions.

start_at

The time the action was started.

end_at

The time the action completed execution.

timeout

The timeout in seconds.

status

Current status of the action.

inputs

A dictionary containing the inputs to the action.

outputs

A dictionary containing the outputs to the action.

depends_on

A list of actions that must finish before this action starts execution.

depended_by

A list of actions that can start only after this action has finished.

created_at

Timestamp when the action is created.

updated_at

Timestamp when the action was last updated.

cluster_id

The ID of cluster which this action runs on.

openstack.clustering.v1.event

The Event Class

The Event class inherits from *Resource*.

class openstack.clustering.v1.event.**Event**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of self._connection in Resource code should protect itself with a check for None.

resource_key = 'event'

Singular form of key for resource.

resources_key = 'events'

Plural form of key for resource.

base_path = '/events'

The base part of the URI for this resource.

allow_list = True

Allow list operation for this resource.

allow_fetch = True

Allow get operation for this resource.

generated_at

Timestamp string (in ISO8601 format) when the event was generated.

obj_id

The UUID of the object related to this event.

obj_name
The name of the object related to this event.

obj_type
The type name of the object related to this event.

cluster_id
The UUID of the cluster related to this event, if any.

level
The event level (priority).

user_id
The ID of the user.

project_id
The ID of the project (tenant).

action
The string representation of the action associated with the event.

status
The status of the associated object.

status_reason
A string description of the reason that brought the object into its current status.

meta_data
The metadata of an event object.

Compute Resources

openstack.compute.v2.aggregate

The Aggregate Class

The `Aggregate` class inherits from *Resource*.

class openstack.compute.v2.aggregate.**Aggregate**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'aggregate'
Singular form of key for resource.

resources_key = 'aggregates'

Plural form of key for resource.

base_path = '/os-aggregates'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_commit = True

Allow update operation for this resource.

availability_zone

Availability zone of aggregate

created_at

The date and time when the resource was created.

deleted_at

The date and time when the resource was deleted.

is_deleted

Deleted?

name

Name of aggregate

hosts

Hosts

metadata

Metadata

updated_at

The date and time when the resource was updated

uuid

UUID

add_host(*session*, *host*)

Adds a host to an aggregate.

remove_host(*session*, *host*)

Removes a host from an aggregate.

set_metadata(*session*, *metadata*)

Creates or replaces metadata for an aggregate.

precache_images(*session*, *images*)

Requests image pre-caching

openstack.compute.v2.availability_zone

The AvailabilityZone Class

The `AvailabilityZone` class inherits from *Resource*.

class openstack.compute.v2.availability_zone.**AvailabilityZone**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'availabilityZoneInfo'

Plural form of key for resource.

base_path = '/os-availability-zone'

The base part of the URI for this resource.

allow_list = True

Allow list operation for this resource.

name

name of availability zone

state

state of availability zone

hosts

hosts of availability zone

openstack.compute.v2.extension

The Extension Class

The `Extension` class inherits from *Resource*.

class openstack.compute.v2.extension.**Extension**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.

- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'extension'

Singular form of key for resource.

resources_key = 'extensions'

Plural form of key for resource.

base_path = '/extensions'

The base part of the URI for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_list = True

Allow list operation for this resource.

alias

A short name by which this extension is also known.

description

Text describing this extensions purpose.

links

Links pertaining to this extension. This is a list of dictionaries, each including keys `href` and `rel`.

name

The name of the extension.

namespace

A URL pointing to the namespace for this extension.

updated_at

Timestamp when this extension was last updated.

openstack.compute.v2.flavor

The Flavor Class

The `Flavor` class inherits from *Resource*.

class openstack.compute.v2.flavor.**Flavor**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be

used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'flavor'

Singular form of key for resource.

resources_key = 'flavors'

Plural form of key for resource.

base_path = '/flavors'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_commit = True

Allow update operation for this resource.

name

The name of this flavor.

original_name

The name of this flavor when returned by server list/show

description

The description of the flavor.

disk

Size of the disk this flavor offers. *Type: int*

is_public

`True` if this is a publicly visible flavor. `False` if this is a private image. *Type: bool*

ram

The amount of RAM (in MB) this flavor offers. *Type: int*

vcpus

The number of virtual CPUs this flavor offers. *Type: int*

swap

Size of the swap partitions.

ephemeral

Size of the ephemeral data disk attached to this server. *Type: int*

is_disabled

`True` if this flavor is disabled, `False` if not. *Type: bool*

rxtx_factor

The bandwidth scaling factor this flavor receives on the network.

extra_specs

A dictionary of the flavors extra-specs key-and-value pairs.

classmethod list(*session*, *paginated=True*, *base_path='/flavors/detail'*, ***params*)

This method is a generator which yields resource objects.

This resource object list generator handles pagination and takes query params for response filtering.

Parameters

- **session** (Adapter) The session to use for making this request.
- **paginated** (*bool*) True if a GET to this resource returns a paginated series of responses, or False if a GET returns only one page of data. **When paginated is False only one page of data will be returned regardless of the APIs support of pagination.**
- **base_path** (*str*) Base part of the URI for listing resources, if different from *base_path*.
- **allow_unknown_params** (*bool*) True to accept, but discard unknown query parameters. This allows getting list of filters and passing everything known to the server. False will result in validation exception when unknown query parameters are passed.
- **microversion** (*str*) API version to override the negotiated one.
- **headers** (*dict*) Additional headers to inject into the HTTP request.
- **params** (*dict*) These keyword arguments are passed through the _transpose() method to find if any of them match expected query parameters to be sent in the *params* argument to get(). They are additionally checked against the *base_path* format string to see if any path fragments need to be filled in by the contents of this argument. Parameters supported as filters by the server side are passed in the API call, remaining parameters are applied as filters to the retrieved results.

Returns

A generator of Resource objects.

Raises

MethodNotSupported if Resource.allow_list is not set to True.

Raises

InvalidResourceQuery if query contains invalid params.

add_tenant_access(*session*, *tenant*)

Adds flavor access to a tenant and flavor.

Parameters

- **session** The session to use for making this request.
- **tenant**

Returns
None

`remove_tenant_access`(*session*, *tenant*)

Removes flavor access to a tenant and flavor.

Parameters

- **session** The session to use for making this request.
- **tenant**

Returns
None

`get_access`(*session*)

Lists tenants who have access to a private flavor

By default, only administrators can manage private flavor access. A private flavor has `is_public` set to false while a public flavor has `is_public` set to true.

Parameters
session The session to use for making this request.

Returns
List of dicts with flavor_id and tenant_id attributes

`fetch_extra_specs`(*session*)

Fetch extra specs of the flavor

Starting with 2.61 extra specs are returned with the flavor details, before that a separate call is required.

Parameters
session The session to use for making this request.

Returns
The updated flavor.

`create_extra_specs`(*session*, *specs*)

Creates extra specs for a flavor.

Parameters

- **session** The session to use for making this request.
- **specs**

Returns
The updated flavor.

`get_extra_specs_property`(*session*, *prop*)

Get an individual extra spec property.

Parameters

- **session** The session to use for making this request.
- **prop** The property to fetch.

Returns
The value of the property if it exists, else `None`.

update_extra_specs_property(*session*, *prop*, *val*)

Update an extra spec for a flavor.

Parameters

- **session** The session to use for making this request.
- **prop** The property to update.
- **val** The value to update with.

Returns

The updated value of the property.

delete_extra_specs_property(*session*, *prop*)

Delete an extra spec for a flavor.

Parameters

- **session** The session to use for making this request.
- **prop** The property to delete.

Returns

None

The FlavorDetail Class

The `FlavorDetail` class inherits from *Flavor*.

`openstack.compute.v2.flavor.`**FlavorDetail**

alias of *Flavor*

openstack.compute.v2.hypervisor

The Hypervisor Class

The `Hypervisor` class inherits from *Resource*.

class `openstack.compute.v2.hypervisor.`**Hypervisor**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (`openstack.connection.Connection`) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'hypervisor'

Singular form of key for resource.

resources_key = 'hypervisors'

Plural form of key for resource.

base_path = '/os-hypervisors'

The base part of the URI for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_list = True

Allow list operation for this resource.

cpu_info

Information about the hypervisors CPU. Up to 2.28 it was string.

host_ip

IP address of the host

hypervisor_type

The type of hypervisor

hypervisor_version

Version of the hypervisor

name

Name of hypervisor

service_details

Service details

servers

List of Servers

state

State of hypervisor

status

Status of hypervisor

uptime

The total uptime of the hypervisor and information about average load. This attribute is set only when querying uptime explicitly.

current_workload

Measurement of the hypervisors current workload

disk_available

Disk space available to the scheduler

local_disk_used

The amount, in gigabytes, of local storage used

local_disk_size

The amount, in gigabytes, of the local storage device

local_disk_free

The amount, in gigabytes, of free space on the local storage device

memory_used

The amount, in megabytes, of memory

memory_size

The amount, in megabytes, of total memory

memory_free

The amount, in megabytes, of available memory

running_vms

Count of the running virtual machines

vcpus_used

Count of the VCPUs in use

vcpus

Count of all VCPUs

get_uptime(*session*)

Get uptime information for the hypervisor

Updates uptime attribute of the hypervisor object

openstack.compute.v2.image

The Image Class

The `Image` class inherits from *Resource*.

class `openstack.compute.v2.image.`**Image**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (`openstack.connection.Connection`) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'image'

Singular form of key for resource.

resources_key = 'images'

Plural form of key for resource.

base_path = '/images'

The base part of the URI for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

links

Links pertaining to this image. This is a list of dictionaries, each including keys `href` and `rel`, and optionally `type`.

name

The name of this image.

created_at

Timestamp when the image was created.

min_disk

The mimimum disk size. *Type: int*

min_ram

The minimum RAM size. *Type: int*

progress

If this image is still building, its progress is represented here. Once an image is created, progres will be 100. *Type: int*

status

The status of this image.

updated_at

Timestamp when the image was updated.

size

Size of the image in bytes. *Type: int*

The ImageDetail Class

The `ImageDetail` class inherits from *Image*.

`openstack.compute.v2.image.`**ImageDetail**

alias of *Image*

openstack.compute.v2.keypair

The Keypair Class

The `Keypair` class inherits from *Resource*.

class `openstack.compute.v2.keypair.`**Keypair**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (`openstack.connection.Connection`) Reference to the Connection being used. Defaults to None to allow Resource objects to be

used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'keypair'

Singular form of key for resource.

resources_key = 'keypairs'

Plural form of key for resource.

base_path = '/os-keypairs'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

created_at

The date and time when the resource was created.

is_deleted

A boolean indicates whether this keypair is deleted or not.

fingerprint

The short fingerprint associated with the `public_key` for this keypair.

id

The id identifying the keypair

name

A name identifying the keypair

private_key

The private key for the keypair

public_key

The SSH public key that is paired with the server.

type

The type of the keypair.

user_id

The user_id for a keypair.

classmethod existing(*connection=None*, ***kwargs*)

Create an instance of an existing remote resource.

When creating the instance set the `_synchronized` parameter of `Resource` to `True` to indicate that it represents the state of an existing server-side resource. As such, all attributes

passed in **kwargs are considered clean, such that an immediate update() call would not generate a body of attributes to be modified on the server.

Parameters
kwargs (*dict*) Each of the named arguments will be set as attributes on the resulting Resource object.

openstack.compute.v2.limits

The Limits Class

The Limits class inherits from *Resource*.

class openstack.compute.v2.limits.**Limits**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of self._connection in Resource code should protect itself with a check for None.

base_path = '/limits'

The base part of the URI for this resource.

resource_key = 'limits'

Singular form of key for resource.

allow_fetch = True

Allow get operation for this resource.

absolute

An absolute limits object.

rate

Rate-limit compute resources. This is only populated when using the legacy v2 API which was removed in Nova 14.0.0 (Newton). In v2.1 it will always be an empty list.

fetch(*session*, *requires_id=False*, *base_path=None*, *error_message=None*, *skip_cache=False*, ***params*)

Get the Limits resource.

Parameters
session (Adapter) The session to use for making this request.

Returns
A Limits instance

Return type
Limits

The AbsoluteLimits Class

The `AbsoluteLimits` class inherits from *Resource*.

class openstack.compute.v2.limits.**AbsoluteLimits**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

image_meta

The number of key-value pairs that can be set as image metadata.

personality

The maximum number of personality contents that can be supplied.

personality_size

The maximum size, in bytes, of a personality.

security_group_rules

The maximum amount of security group rules allowed.

security_groups

The maximum amount of security groups allowed.

security_groups_used

The amount of security groups currently in use.

server_meta

The number of key-value pairs that can be set as server metadata.

total_cores

The maximum amount of cores.

total_cores_used

The amount of cores currently in use.

floating_ips

The maximum amount of floating IPs.

floating_ips_used

The amount of floating IPs currently in use.

instances

The maximum amount of instances.

instances_used

The amount of instances currently in use.

keypairs
The maximum amount of keypairs.

total_ram
The maximum RAM size in megabytes.

total_ram_used
The RAM size in megabytes currently in use.

server_groups
The maximum amount of server groups.

server_groups_used
The amount of server groups currently in use.

server_group_members
The maximum number of members in a server group.

The RateLimit Class

The `RateLimit` class inherits from *Resource*.

class openstack.compute.v2.limits.**RateLimit**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

next_available
Rate limits next availabe time.

remaining
Integer for rate limits remaining.

unit
Unit of measurement for the value parameter.

value
Integer number of requests which can be made.

verb
An HTTP verb (POST, PUT, etc.).

openstack.compute.v2.migration

The Migration Class

The `Migration` class inherits from *Resource*.

class openstack.compute.v2.migration.**Migration**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'migrations'

Plural form of key for resource.

base_path = '/os-migrations'

The base part of the URI for this resource.

allow_list = True

Allow list operation for this resource.

created_at

The date and time when the resource was created.

dest_compute

The target compute of the migration.

dest_host

The target host of the migration.

dest_node

The target node of the migration.

migration_type

The type of the migration. One of migration, resize, live-migration or evacuation

new_flavor_id

The ID of the old flavor. This value corresponds to the ID of the flavor in the database. This will be the same as new_flavor_id except for resize operations.

old_flavor_id

The ID of the old flavor. This value corresponds to the ID of the flavor in the database.

project_id

The ID of the project that initiated the server migration (since microversion 2.80)

server_id

The UUID of the server

source_compute

The source compute of the migration.

source_node

The source node of the migration.

status

The current status of the migration.

updated_at

The date and time when the resource was last updated.

user_id

The ID of the user that initiated the server migration (since microversion 2.80)

uuid

The UUID of the migration (since microversion 2.59)

openstack.compute.v2.quota_set

The QuotaSet Class

The QuotaSet class inherits from *Resource*.

class openstack.compute.v2.quota_set.**QuotaSet**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of self._connection in Resource code should protect itself with a check for None.

cores

Properties The number of allowed server cores for each tenant.

fixed_ips

The number of allowed fixed IP addresses for each tenant. Must be equal to or greater than the number of allowed servers.

floating_ips

The number of allowed floating IP addresses for each tenant.

force

You can force the update even if the quota has already been used and the reserved quota exceeds the new quota.

injected_file_content_bytes

The number of allowed bytes of content for each injected file.

injected_file_path_bytes

The number of allowed bytes for each injected file path.

injected_files

The number of allowed injected files for each tenant.

instances

The number of allowed servers for each tenant.

key_pairs

The number of allowed key pairs for each user.

metadata_items

The number of allowed metadata items for each server.

networks

The number of private networks that can be created per project.

ram

The amount of allowed server RAM, in MiB, for each tenant.

security_group_rules

The number of allowed rules for each security group.

security_groups

The number of allowed security groups for each tenant.

server_groups

The number of allowed server groups for each tenant.

server_group_members

The number of allowed members for each server group.

openstack.compute.v2.server

The Server Class

The Server class inherits from *Resource*.

class openstack.compute.v2.server.**Server**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of self._connection in Resource code should protect itself with a check for None.

resource_key = 'server'

Singular form of key for resource.

resources_key = 'servers'

Plural form of key for resource.

base_path = '/servers'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

links

A list of dictionaries holding links relevant to this server.

addresses

A dictionary of addresses this server can be accessed through. The dictionary contains keys such as `private` and `public`, each containing a list of dictionaries for addresses of that type. The addresses are contained in a dictionary with keys `addr` and `version`, which is either 4 or 6 depending on the protocol of the IP address. *Type: dict*

admin_password

When a server is first created, it provides the administrator password.

attached_volumes

A list of an attached volumes. Each item in the list contains at least an id key to identify the specific volumes.

availability_zone

The name of the availability zone this server is a part of.

block_device_mapping

Enables fine grained control of the block device mapping for an instance. This is typically used for booting servers from volumes.

config_drive

Indicates whether or not a config drive was used for this server.

compute_host

The name of the compute host on which this instance is running. Appears in the response for administrative users only.

created_at

Timestamp of when the server was created.

description

The description of the server. Before microversion 2.19 this was set to the server name.

disk_config

The disk configuration. Either AUTO or MANUAL.

flavor_id

The flavor reference, as a ID or full URL, for the flavor to use for this server.

flavor

The flavor property as returned from server.

has_config_drive

Indicates whether a configuration drive enables metadata injection. Not all cloud providers enable this feature.

host_id

An ID representing the host of this server.

fault

A fault object. Only available when the server status is ERROR or DELETED and a fault occurred.

host

The host to boot the server on.

host_status

The host status.

hostname

The hostname set on the instance when it is booted. By default, it appears in the response for administrative users only.

hypervisor_hostname

The hypervisor host name. Appears in the response for administrative users only.

image_id

The image reference, as a ID or full URL, for the image to use for this server.

image

The image property as returned from server.

instance_name

The instance name. The Compute API generates the instance name from the instance name template. Appears in the response for administrative users only.

interface_ip

The address to use to connect to this server from the current calling context. This will be set to public_ipv6 if the calling host has routable ipv6 addresses, and to private_ipv4 if the Connection was created with private=True. Otherwise it will be set to public_ipv4.

kernel_id

The UUID of the kernel image when using an AMI. Will be null if not. By default, it appears in the response for administrative users only.

key_name

The name of an associated keypair

launch_index

When servers are launched via multiple create, this is the sequence in which the servers were launched. By default, it appears in the response for administrative users only.

launched_at

The timestamp when the server was launched.

locked_reason

The reason the server was locked, if any.

max_count

The maximum number of servers to create.

min_count

The minimum number of servers to create.

networks

A networks object. Required parameter when there are multiple networks defined for the tenant. When you do not specify the networks parameter, the server attaches to the only network created for the current tenant.

personality

Personality files. This should be a list of dicts with each dict containing a file name (name) and a base64-encoded file contents (contents)

pinned_availability_zone

The availability zone requested during server creation OR pinned availability zone, which is configured using default_schedule_zone config option.

power_state

The power state of this server.

progress

While the server is building, this value represents the percentage of completion. Once it is completed, it will be 100. *Type: int*

project_id

The ID of the project this server is associated with.

private_v4

The private IPv4 address of this server

private_v6

The private IPv6 address of this server

public_v4

The public IPv4 address of this server

public_v6

The public IPv6 address of this server

ramdisk_id

The UUID of the ramdisk image when using an AMI. Will be null if not. By default, it appears in the response for administrative users only.

reservation_id

The reservation id for the server. This is an id that can be useful in tracking groups of servers created with multiple create, that will all have the same reservation_id. By default, it appears in the response for administrative users only.

root_device_name

The root device name for the instance By default, it appears in the response for administrative users only.

scheduler_hints

The dictionary of data to send to the scheduler.

security_groups

A list of applicable security groups. Each group contains keys for description, name, id, and rules.

server_groups

The UUIDs of the server groups to which the server belongs. Currently this can contain at most one entry.

status

The state this server is in. Valid values include ACTIVE, BUILDING, DELETED, ERROR, HARD_REBOOT, PASSWORD, PAUSED, REBOOT, REBUILD, RESCUED, RESIZED, REVERT_RESIZE, SHUTOFF, SOFT_DELETED, STOPPED, SUSPENDED, UNKNOWN, or VERIFY_RESIZE.

task_state

The task state of this server.

terminated_at

The timestamp when the server was terminated (if it has been).

trusted_image_certificates

A list of trusted certificate IDs, that were used during image signature verification to verify the signing certificate.

updated_at

Timestamp of when this server was last updated.

user_data

Configuration information or scripts to use upon launch. Must be Base64 encoded.

user_id

The ID of the owners of this server.

vm_state

The VM state of this server.

change_password(*session*, *password*, *, *microversion=None*)

Change the administrator password to the given password.

Parameters

- **session** The session to use for making this request.
- **password** The new password.

Returns

None

get_password(*session*, *, *microversion=None*)

Get the encrypted administrator password.

Parameters

session The session to use for making this request.

Returns
The encrypted administrator password.

clear_password(*session*, *, *microversion=None*)

Clear the administrator password.

This removes the password from the database. It does not actually change the server password.

Parameters
session The session to use for making this request.

Returns
None

reboot(*session*, *reboot_type*)

Reboot server where reboot_type might be SOFT or HARD.

Parameters

- **session** The session to use for making this request.
- **reboot_type** The type of reboot. One of: `SOFT`, `HARD`.

Returns
None

force_delete(*session*)

Force delete the server.

Parameters
session The session to use for making this request.

Returns
None

rebuild(*session*, *image*, *name=<openstack.compute.v2.server.Unset object>*, *admin_password=<openstack.compute.v2.server.Unset object>*, *preserve_ephemeral=<openstack.compute.v2.server.Unset object>*, *access_ipv4=<openstack.compute.v2.server.Unset object>*, *access_ipv6=<openstack.compute.v2.server.Unset object>*, *metadata=<openstack.compute.v2.server.Unset object>*, *user_data=<openstack.compute.v2.server.Unset object>*, *key_name=<openstack.compute.v2.server.Unset object>*, *description=<openstack.compute.v2.server.Unset object>*, *trusted_image_certificates=<openstack.compute.v2.server.Unset object>*, *hostname=<openstack.compute.v2.server.Unset object>*)

Rebuild the server with the given arguments.

Parameters

- **session** The session to use for making this request.
- **image** The image to rebuild to. Either an ID or a *Image* instance.
- **name** A name to set on the rebuilt server. (Optional)
- **admin_password** An admin password to set on the rebuilt server. (Optional)

- **preserve_ephemeral** Whether to preserve the ephemeral drive during the rebuild. (Optional)
- **access_ipv4** An IPv4 address that will be used to access the rebuilt server. (Optional)
- **access_ipv6** An IPv6 address that will be used to access the rebuilt server. (Optional)
- **metadata** Metadata to set on the updated server. (Optional)
- **user_data** User data to set on the updated server. (Optional)
- **key_name** A key name to set on the updated server. (Optional)
- **description** The description to set on the updated server. (Optional) (Requires API microversion 2.19)
- **trusted_image_certificates** The trusted image certificates to set on the updated server. (Optional) (Requires API microversion 2.78)
- **hostname** The hostname to set on the updated server. (Optional) (Requires API microversion 2.90)

Returns
The updated server.

resize(*session*, *flavor*)

Resize server to flavor reference.

Parameters

- **session** The session to use for making this request.
- **flavor** The server to resize to.

Returns
None

confirm_resize(*session*)

Confirm the resize of the server.

Parameters
session The session to use for making this request.

Returns
None

revert_resize(*session*)

Revert the resize of the server.

Parameters
session The session to use for making this request.

Returns
None

create_image(*session*, *name*, *metadata=None*)

Create image from server.

Parameters

- **session** The session to use for making this request.
- **name** The name to use for the created image.
- **metadata** Metadata to set on the created image. (Optional)

Returns
None

add_security_group(*session*, *security_group_name*)

Add a security group to the server.

Parameters

- **session** The session to use for making this request.
- **security_group_name** The security group to add to the server.

Returns
None

remove_security_group(*session*, *security_group_name*)

Remove a security group from the server.

Parameters

- **session** The session to use for making this request.
- **security_group_name** The security group to remove from the server.

Returns
None

reset_state(*session*, *state*)

Reset server state.

This is admin-only by default.

Parameters

- **session** The session to use for making this request.
- **state** The state to set on the server.

Returns
None

add_fixed_ip(*session*, *network_id*)

Add a fixed IP to the server.

This is effectively an alias for adding a network.

Parameters

- **session** The session to use for making this request.
- **network_id** The network to connect the server to.

Returns
None

`remove_fixed_ip`(*session*, *address*)

Remove a fixed IP from the server.

This is effectively an alias from removing a port from the server.

Parameters

- **session** The session to use for making this request.
- **network_id** The address to remove from the server.

Returns
None

`add_floating_ip`(*session*, *address*, *fixed_address=None*)

Add a floating IP to the server.

Parameters

- **session** The session to use for making this request.
- **address** The floating IP address to associate with the server.
- **fixed_address** A fixed IP address with which to associated the floating IP. (Optional)

Returns
None

`remove_floating_ip`(*session*, *address*)

Remove a floating IP from the server.

Parameters

- **session** The session to use for making this request.
- **address** The floating IP address to disassociate from the server.

Returns
None

`backup`(*session*, *name*, *backup_type*, *rotation*)

Create a backup of the server.

Parameters

- **session** The session to use for making this request.
- **name** The name to use for the backup image.
- **backup_type** The type of backup. The value and meaning of this atribute is user-defined and can be used to separate backups of different types. For example, this could be used to distinguish between `daily` and `weekly` backups.
- **rotation** The number of backups to retain. All images older than the rotationth image will be deleted.

Returns
None

pause(*session*)

Pause the server.

Parameters
session The session to use for making this request.

Returns
None

unpause(*session*)

Unpause the server.

Parameters
session The session to use for making this request.

Returns
None

suspend(*session*)

Suspend the server.

Parameters
session The session to use for making this request.

Returns
None

resume(*session*)

Resume the server.

Parameters
session The session to use for making this request.

Returns
None

lock(*session*, *locked_reason=None*)

Lock the server.

Parameters

- **session** The session to use for making this request.
- **locked_reason** The reason for locking the server.

Returns
None

unlock(*session*)

Unlock the server.

Parameters
session The session to use for making this request.

Returns
None

rescue(*session*, *admin_pass=None*, *image_ref=None*)

Rescue the server.

This is admin-only by default.

Parameters

- **session** The session to use for making this request.
- **admin_pass** A new admin password to set on the rescued server. (Optional)
- **image_ref** The image to use when rescuing the server. If not provided, the server will use the existing image. (Optional)

Returns
None

unrescue(*session*)

Unrescue the server.

This is admin-only by default.

Parameters
session The session to use for making this request.

Returns
None

evacuate(*session*, *host=None*, *admin_pass=None*, *force=None*, *on_shared_storage=None*)

Evacuate the server.

Parameters

- **session** The session to use for making this request.
- **host** The host to evacuate the instance to. (Optional)
- **admin_pass** The admin password to set on the evacuated instance. (Optional)
- **force** Whether to force evacuation.
- **on_shared_storage** Whether the host is using shared storage. (Optional) (Only supported before microversion 2.14)

Returns
None

start(*session*)

Start the server.

Parameters
session The session to use for making this request.

Returns
None

stop(*session*)

Stop the server.

Parameters
session The session to use for making this request.

Returns
None

restore(*session*)

Restore the server.

This is only supported if the server is soft-deleted. This is cloud-specific.

Parameters
session The session to use for making this request.

Returns
None

shelve(*session*)

Shelve the server.

Parameters
session The session to use for making this request.

Returns
None

shelve_offload(*session*)

Shelve-offload the server.

Parameters
session The session to use for making this request.

Returns
None

unshelve(*session*, *availability_zone=<object object>*, *host=None*)

Unshelve the server.

Parameters

- **session** The session to use for making this request.
- **availability_zone** If specified the instance will be unshelved to the availability_zone. If None is passed the instance defined availability_zone is unpin and the instance will be scheduled to any availability_zone (free scheduling). If not specified the instance will be unshelved to either its defined availability_zone or any availability_zone (free scheduling).
- **host** If specified the host to unshelve the instance.

migrate(*session*, *, *host=None*)

Migrate the server.

Parameters

- **session** The session to use for making this request.
- **host** The host to migrate the server to. (Optional)

Returns
None

trigger_crash_dump(*session*)

Trigger a crash dump for the server.

Parameters
session The session to use for making this request.

Returns
None

`get_console_output`(*session*, *length=None*)

Get console output for the server.

Parameters

- **session** The session to use for making this request.
- **length** The max length of the console output to return. (Optional)

Returns
None

`get_console_url`(*session*, *console_type*)

Get the console URL for the server.

Parameters

- **session** The session to use for making this request.
- **console_type** The type of console to return. This is cloud-specific. One of: `novnc`, `xvpvnc`, `spice-html5`, `rdp-html5`, `serial`.

Returns
None

`live_migrate`(*session*, *host*, *force*, *block_migration*, *disk_over_commit=False*)

Live migrate the server.

Parameters

- **session** The session to use for making this request.
- **host** The host to live migrate the server to. (Optional)
- **force** Whether to force the migration. (Optional)
- **block_migration** Whether to do block migration. One of: `True`, `False`, `'auto'`. (Optional)
- **disk_over_commit** Whether to allow disk over-commit on the destination host. (Optional)

Returns
None

`fetch_topology`(*session*)

Fetch the topology information for the server.

Parameters
session The session to use for making this request.

Returns
None

`fetch_security_groups`(*session*)

Fetch security groups of the server.

Parameters
session The session to use for making this request.

Returns
Updated Server instance.

openstack.compute.v2.server_action

The ServerAction Class

The `ServerAction` class inherits from *Resource*.

class openstack.compute.v2.server_action.**ServerAction**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'instanceAction'
Singular form of key for resource.

resources_key = 'instanceActions'
Plural form of key for resource.

base_path = '/servers/%(server_id)s/os-instance-actions'
The base part of the URI for this resource.

allow_fetch = True
Allow get operation for this resource.

allow_list = True
Allow list operation for this resource.

server_id
The ID of the server that this action relates to.

action
The name of the action.

request_id
The ID of the request that this action related to.

user_id
The ID of the user which initiated the server action.

project_id
The ID of the project that this server belongs to.

message
The related error message for when an action fails.

events
Events

The ServerActionEvent Class

The `ServerActionEvent` class inherits from *Resource*.

class openstack.compute.v2.server_action.**ServerActionEvent**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

event

The name of the event

start_time

The date and time when the event was started. The date and time stamp format is ISO 8601

finish_time

The date and time when the event finished. The date and time stamp format is ISO 8601

result

The result of the event

traceback

The traceback stack if an error occurred in this event. This is only visible to cloud admins by default.

host

The name of the host on which the event occurred. This is only visible to cloud admins by default.

host_id

An obfuscated hashed host ID string, or the empty string if there is no host for the event. This is a hashed value so will not actually look like a hostname, and is hashed with data from the project_id, so the same physical host as seen by two different project_ids will be different. This is useful when within the same project you need to determine if two events occurred on the same or different physical hosts.

details

Details of the event. May be unset.

openstack.compute.v2.server_diagnostics

The ServerDiagnostics Class

The `ServerDiagnostics` class inherits from *Resource*.

class openstack.compute.v2.server_diagnostics.**ServerDiagnostics**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'diagnostics'

Singular form of key for resource.

base_path = '/servers/%(server_id)s/diagnostics'

The base part of the URI for this resource.

allow_fetch = True

Allow get operation for this resource.

requires_id = False

Do calls for this resource require an id

has_config_drive

Indicates whether or not a config drive was used for this server.

state

The current state of the VM.

driver

The driver on which the VM is running.

hypervisor

The hypervisor on which the VM is running.

hypervisor_os

The hypervisor OS.

uptime

The amount of time in seconds that the VM has been running.

num_cpus

The number of vCPUs.

num_disks

The number of disks.

num_nics

The number of vNICs.

memory_details

The dictionary with information about VM memory usage.

cpu_details
The list of dictionaries with detailed information about VM CPUs.

disk_details
The list of dictionaries with detailed information about VM disks.

nic_details
The list of dictionaries with detailed information about VM NICs.

server_id
The ID for the server.

openstack.compute.v2.server_group

The ServerGroup Class

The ServerGroup class inherits from *Resource*.

class openstack.compute.v2.server_group.**ServerGroup**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of self._connection in Resource code should protect itself with a check for None.

resource_key = 'server_group'
Singular form of key for resource.

resources_key = 'server_groups'
Plural form of key for resource.

base_path = '/os-server-groups'
The base part of the URI for this resource.

allow_create = True
Allow create operation for this resource.

allow_fetch = True
Allow get operation for this resource.

allow_delete = True
Allow delete operation for this resource.

allow_list = True
Allow list operation for this resource.

name
A name identifying the server group

policies

The list of policies supported by the server group (till 2.63)

policy

The policy field represents the name of the policy (from 2.64)

member_ids

The list of members in the server group

metadata

The metadata associated with the server group. This is always empty and only used for preserving compatibility.

project_id

The project ID who owns the server group.

rules

The rules field, which is a dict, can be applied to the policy. Currently, only the max_server_per_host rule is supported for the anti-affinity policy. The max_server_per_host rule allows specifying how many members of the anti-affinity group can reside on the same compute host. If not specified, only one member from the same anti-affinity group can reside on a given host.

user_id

The user ID who owns the server group

create(*session*, *prepend_key=True*, *base_path=None*, ***params*)

Create a remote resource based on this instance.

Parameters

- **session** (Adapter) The session to use for making this request.
- **prepend_key** A boolean indicating whether the resource_key should be prepended in a resource creation request. Default to True.
- **base_path** (*str*) Base part of the URI for creating resources, if different from *base_path*.
- **params** (*dict*) Additional params to pass.

Returns

This Resource instance.

Raises

MethodNotSupported if Resource.allow_create is not set to True.

openstack.compute.v2.server_interface

The ServerInterface Class

The ServerInterface class inherits from *Resource*.

class openstack.compute.v2.server_interface.**ServerInterface**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'interfaceAttachment'

Singular form of key for resource.

resources_key = 'interfaceAttachments'

Plural form of key for resource.

base_path = '/servers/%(server_id)s/os-interface'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = False

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

fixed_ips

Fixed IP addresses with subnet IDs.

mac_addr

The MAC address.

net_id

The network ID.

port_id

The ID of the port for which you want to create an interface.

port_state

The port state.

server_id

The ID for the server.

tag

Tags for the virtual interfaces.

openstack.compute.v2.server_ip

The ServerIP Class

The ServerIP class inherits from *Resource*.

class openstack.compute.v2.server_ip.**ServerIP**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'addresses'

Plural form of key for resource.

base_path = '/servers/%(server_id)s/ips'

The base part of the URI for this resource.

allow_list = True

Allow list operation for this resource.

address

The IP address. The format of the address depends on `version`

network_label

The network label, such as public or private.

server_id

The ID for the server.

classmethod list(*session*, *paginated=False*, *server_id=None*, *network_label=None*, *base_path=None*, ***params*)

This method is a generator which yields resource objects.

This resource object list generator handles pagination and takes query params for response filtering.

Parameters

- **session** (Adapter) The session to use for making this request.
- **paginated** (*bool*) `True` if a GET to this resource returns a paginated series of responses, or `False` if a GET returns only one page of data. **When paginated is False only one page of data will be returned regardless of the APIs support of pagination.**
- **base_path** (*str*) Base part of the URI for listing resources, if different from *base_path*.

- **allow_unknown_params** (*bool*) `True` to accept, but discard unknown query parameters. This allows getting list of filters and passing everything known to the server. `False` will result in validation exception when unknown query parameters are passed.
- **microversion** (*str*) API version to override the negotiated one.
- **headers** (*dict*) Additional headers to inject into the HTTP request.
- **params** (*dict*) These keyword arguments are passed through the `_transpose()` method to find if any of them match expected query parameters to be sent in the *params* argument to `get()`. They are additionally checked against the *base_path* format string to see if any path fragments need to be filled in by the contents of this argument. Parameters supported as filters by the server side are passed in the API call, remaining parameters are applied as filters to the retrieved results.

Returns
A generator of `Resource` objects.

Raises
MethodNotSupported if `Resource.allow_list` is not set to `True`.

Raises
InvalidResourceQuery if query contains invalid params.

openstack.compute.v2.server_migration

The ServerMigration Class

The `ServerMigration` class inherits from *Resource*.

class openstack.compute.v2.server_migration.**ServerMigration**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'migration'

Singular form of key for resource.

resources_key = 'migrations'

Plural form of key for resource.

base_path = '/servers/%(server_id)s/migrations'

The base part of the URI for this resource.

`allow_fetch = True`

Allow get operation for this resource.

`allow_list = True`

Allow list operation for this resource.

`allow_delete = True`

Allow delete operation for this resource.

`server_id`

The ID for the server from the URI of the resource

`created_at`

The date and time when the resource was created.

`dest_host`

The target host of the migration.

`dest_compute`

The target compute of the migration.

`dest_node`

The target node of the migration.

`disk_processed_bytes`

The amount of disk, in bytes, that has been processed during the migration.

`disk_remaining_bytes`

The amount of disk, in bytes, that still needs to be migrated.

`disk_total_bytes`

The total amount of disk, in bytes, that needs to be migrated.

`memory_processed_bytes`

The amount of memory, in bytes, that has been processed during the migration.

`memory_remaining_bytes`

The amount of memory, in bytes, that still needs to be migrated.

`memory_total_bytes`

The total amount of memory, in bytes, that needs to be migrated.

`project_id`

The ID of the project that initiated the server migration (since microversion 2.80)

`server_uuid`

The UUID of the server from the response body

`source_compute`

The source compute of the migration.

`source_node`

The source node of the migration.

`status`

The current status of the migration.

updated_at

The date and time when the resource was last updated.

user_id

The ID of the user that initiated the server migration (since microversion 2.80)

uuid

The UUID of the migration (since microversion 2.59)

force_complete(*session*)

Force on-going live migration to complete.

openstack.compute.v2.server_remote_console

The ServerRemoteConsole Class

The `ServerRemoteConsole` class inherits from *Resource*.

class openstack.compute.v2.server_remote_console.**ServerRemoteConsole**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'remote_console'

Singular form of key for resource.

base_path = '/servers/%(server_id)s/remote-consoles'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = False

Allow get operation for this resource.

allow_commit = False

Allow update operation for this resource.

allow_delete = False

Allow delete operation for this resource.

allow_list = False

Allow list operation for this resource.

protocol
Protocol of the remote console.

type
Type of the remote console.

url
URL used to connect to the console.

server_id
The ID for the server.

create(*session*, *prepend_key=True*, *base_path=None*, ***params*)
Create a remote resource based on this instance.

Parameters

- **session** (Adapter) The session to use for making this request.
- **prepend_key** A boolean indicating whether the resource_key should be prepended in a resource creation request. Default to True.
- **base_path** (*str*) Base part of the URI for creating resources, if different from *base_path*.
- **resource_request_key** (*str*) Overrides the usage of self.resource_key when prepending a key to the request body. Ignored if *prepend_key* is false.
- **resource_response_key** (*str*) Overrides the usage of self.resource_key when processing response bodies. Ignored if *prepend_key* is false.
- **microversion** (*str*) API version to override the negotiated one.
- **params** (*dict*) Additional params to pass.

Returns
This Resource instance.

Raises
MethodNotSupported if Resource.allow_create is not set to True.

openstack.compute.v2.service

The Service Class

The Service class inherits from *Resource*.

class openstack.compute.v2.service.**Service**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of self._connection in Resource code should protect itself with a check for None.

resource_key = 'service'

Singular form of key for resource.

resources_key = 'services'

Plural form of key for resource.

base_path = '/os-services'

The base part of the URI for this resource.

allow_list = True

Allow list operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

availability_zone

The availability zone of service

binary

Binary name of service

disabled_reason

Disabled reason of service

is_forced_down

Whether or not this service was forced down manually by an administrator after the service was fenced

host

The name of the host where service runs

name

Service name

state

State of service

status

Status of service

updated_at

The date and time when the resource was updated

classmethod find(*session*, *name_or_id*, *ignore_missing=True*, ***params*)

Find a resource by its name or id.

Parameters

- **session** (Adapter) The session to use for making this request.
- **name_or_id** This resources identifier, if needed by the request. The default is None.

- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **list_base_path** (*str*) base_path to be used when need listing resources.
- **microversion** (*str*) API version to override the negotiated one.
- **params** (*dict*) Any additional parameters to be passed into underlying methods, such as to *existing()* in order to pass on URI parameters.

Returns
The `Resource` object matching the given name or id or None if nothing matches.

Raises
openstack.exceptions.DuplicateResource if more than one resource is found for this request.

Raises
openstack.exceptions.NotFoundException if nothing is found and ignore_missing is `False`.

commit(*session*, *prepend_key=False*, **args*, ***kwargs*)

Commit the state of the instance to the remote resource.

Parameters

- **session** (`Adapter`) The session to use for making this request.
- **prepend_key** A boolean indicating whether the resource_key should be prepended in a resource update request. Default to True.
- **retry_on_conflict** (*bool*) Whether to enable retries on HTTP CONFLICT (409). Value of `None` leaves the *Adapter* defaults.
- **base_path** (*str*) Base part of the URI for modifying resources, if different from *base_path*.
- **microversion** (*str*) API version to override the negotiated one.
- **kwargs** (*dict*) Parameters that will be passed to _prepare_request()

Returns
This `Resource` instance.

Raises
MethodNotSupported if `Resource.allow_commit` is not set to `True`.

set_forced_down(*session*, *host=None*, *binary=None*, *forced=False*)

Update forced_down information of a service.

force_down(*session*, *host=None*, *binary=None*, *forced=False*)

Update forced_down information of a service.

enable(*session*, *host*, *binary*)

Enable service.

disable(*session*, *host*, *binary*, *reason=None*)

Disable service.

openstack.compute.v2.usage

The Usage Class

The Usage class inherits from *Resource*.

class openstack.compute.v2.usage.**Usage**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of self._connection in Resource code should protect itself with a check for None.

resource_key = 'tenant_usage'

Singular form of key for resource.

resources_key = 'tenant_usages'

Plural form of key for resource.

base_path = '/os-simple-tenant-usage'

The base part of the URI for this resource.

allow_create = False

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_delete = False

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_commit = False

Allow update operation for this resource.

project_id

The UUID of the project in a multi-tenancy cloud.

server_usages

A list of the server usage objects.

total_local_gb_usage

Multiplying the server disk size (in GiB) by hours the server exists, and then adding that all together for each server.

total_vcpus_usage

Multiplying the number of virtual CPUs of the server by hours the server exists, and then adding that all together for each server.

total_memory_mb_usage

Multiplying the server memory size (in MiB) by hours the server exists, and then adding that all together for each server.

total_hours

The total duration that servers exist (in hours).

start

The beginning time to calculate usage statistics on compute and storage resources.

stop

The ending time to calculate usage statistics on compute and storage resources.

The ServerUsage Class

The `ServerUsage` class inherits from *Resource*.

class openstack.compute.v2.usage.**ServerUsage**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = None

Singular form of key for resource.

resources_key = None

Plural form of key for resource.

allow_create = False

Allow create operation for this resource.

allow_fetch = False

Allow get operation for this resource.

allow_delete = False

Allow delete operation for this resource.

allow_list = False

Allow list operation for this resource.

allow_commit = False

Allow update operation for this resource.

hours

The duration that the server exists (in hours).

flavor
The display name of a flavor.

instance_id
The UUID of the server.

name
The server name.

project_id
The UUID of the project in a multi-tenancy cloud.

memory_mb
The memory size of the server (in MiB).

local_gb
The sum of the root disk size of the server and the ephemeral disk size of it (in GiB).

vcpus
The number of virtual CPUs that the server uses.

started_at
The date and time when the server was launched.

ended_at
The date and time when the server was deleted.

state
The VM state.

uptime
The uptime of the server.

openstack.compute.v2.volume_attachment

The VolumeAttachment Class

The `VolumeAttachment` class inherits from *Resource*.

class openstack.compute.v2.volume_attachment.**VolumeAttachment**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'volumeAttachment'
Singular form of key for resource.

resources_key = 'volumeAttachments'

Plural form of key for resource.

base_path = '/servers/%(server_id)s/os-volume_attachments'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

server_id

The ID for the server.

device

Name of the device such as, /dev/vdb.

id

The ID of the attachment.

volume_id

The ID of the attached volume.

attachment_id

The UUID of the associated volume attachment in Cinder.

bdm_id

The ID of the block device mapping record for the attachment

tag

Virtual device tags for the attachment.

delete_on_termination

Indicates whether to delete the volume when server is destroyed

openstack.compute.version

The Version Class

The `Version` class inherits from *Resource*.

class openstack.compute.version.**Version**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'version'

Singular form of key for resource.

resources_key = 'versions'

Plural form of key for resource.

base_path = '/'

The base part of the URI for this resource.

allow_list = True

Allow list operation for this resource.

Container Infrastructure Management Resources

openstack.container_infrastructure_management.v1.cluster

The Cluster Class

The `Cluster` class inherits from *Resource*.

class openstack.container_infrastructure_management.v1.cluster.**Cluster**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'clusters'

Plural form of key for resource.

base_path = '/clusters'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_patch = True

Allow patch operation for this resource.

commit_method = 'PATCH'

Method for committing a resource (PUT, PATCH, POST)

commit_jsonpatch = True

Whether commit uses JSON patch format.

api_address

The endpoint URL of COE API exposed to end-users.

cluster_template_id

The UUID of the cluster template.

coe_version

Version info of chosen COE in bay/cluster for helping client in picking the right version of client.

create_timeout

The timeout for cluster creation in minutes. The value expected is a positive integer. If the timeout is reached during cluster creation process, the operation will be aborted and the cluster status will be set to CREATE_FAILED. Defaults to 60.

created_at

The date and time when the resource was created. The date and time stamp format is ISO 8601:

```
CCYY-MM-DDThh:mm:ssśhh:mm
```

For example, *2015-08-27T09:49:58-05:00*. The śhh:mm value, if included, is the time zone as an offset from UTC.

discovery_url

The custom discovery url for node discovery. This is used by the COE to discover the servers that have been created to host the containers. The actual discovery mechanism varies with the COE. In some cases, the service fills in the server info in the discovery service. In other cases,if the discovery_url is not specified, the service will use the public discovery service at https://discovery.etcd.io. In this case, the service will generate a unique url here for each bay and store the info for the servers.

fixed_network

The name or ID of the network to provide connectivity to the internal network for the bay/cluster.

fixed_subnet

The fixed subnet to use when allocating network addresses for nodes in bay/cluster.

flavor_id

The flavor name or ID to use when booting the node servers. Defaults to m1.small.

is_floating_ip_enabled

Whether to enable using the floating IP of cloud provider. Some cloud providers use floating IPs while some use public IPs. When set to true, floating IPs will be used. If this value is not provided, the value of `floating_ip_enabled` provided in the template will be used.

is_master_lb_enabled

Whether to enable the master load balancer. Since multiple masters may exist in a bay/cluster, a Neutron load balancer is created to provide the API endpoint for the bay/cluster and to direct requests to the masters. In some cases, such as when the LBaaS service is not available, this option can be set to false to create a bay/cluster without the load balancer. In this case, one of the masters will serve as the API endpoint. The default is true, i.e. to create the load balancer for the bay.

keypair

The name of the SSH keypair to configure in the bay/cluster servers for SSH access. Users will need the key to be able to ssh to the servers in the bay/cluster. The login name is specific to the bay/cluster driver. For example, with fedora-atomic image the default login name is fedora.

labels

Arbitrary labels. The accepted keys and valid values are defined in the bay/cluster drivers. They are used as a way to pass additional parameters that are specific to a bay/cluster driver.

master_addresses

A list of floating IPs of all master nodes.

master_count

The number of servers that will serve as master for the bay/cluster. Set to more than 1 master to enable High Availability. If the option master-lb-enabled is specified in the baymodel/cluster template, the master servers will be placed in a load balancer pool. Defaults to 1.

master_flavor_id

The flavor of the master node for this baymodel/cluster template.

name

Name of the resource.

node_count

The number of servers that will serve as node in the bay/cluster. Defaults to 1.

node_addresses

A list of floating IPs of all servers that serve as nodes.

stack_id

The reference UUID of orchestration stack from Heat orchestration service.

status

The current state of the bay/cluster.

status_reason

The reason of bay/cluster current status.

updated_at

The date and time when the resource was updated. The date and time stamp format is ISO 8601:

```
CCYY-MM-DDThh:mm:ssśhh:mm
```

For example, *2015-08-27T09:49:58-05:00*. The śhh:mm value, if included, is the time zone as an offset from UTC. If the updated_at date and time stamp is not set, its value is null.

uuid

The UUID of the cluster.

resize(*session*, ***, *node_count*, *nodes_to_remove=None*)

Resize the cluster.

Parameters

- **node_count** The number of servers that will serve as node in the bay/cluster. The default is 1.
- **nodes_to_remove** The server ID list will be removed if downsizing the cluster.

Returns

The UUID of the resized cluster.

Raises

NotFoundException if the resource was not found.

upgrade(*session*, ***, *cluster_template*, *max_batch_size=None*)

Upgrade the cluster.

Parameters

- **cluster_template** The UUID of the cluster template.
- **max_batch_size** The max batch size each time when doing upgrade. The default is 1

Returns

The UUID of the updated cluster.

Raises

NotFoundException if the resource was not found.

openstack.container_infrastructure_management.v1.cluster_certificate

The Cluster Certificate Class

The `ClusterCertificate` class inherits from *Resource*.

class openstack.container_infrastructure_management.v1.cluster_certificate.ClusterCertificat

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

base_path = '/certificates'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_list = False

Allow list operation for this resource.

allow_fetch = True

Allow get operation for this resource.

bay_uuid

The UUID of the bay.

cluster_uuid

The UUID of the cluster.

csr

Certificate Signing Request (CSR) for authenticating client key.

pem

CA certificate for the bay/cluster.

openstack.container_infrastructure_management.v1.cluster_template

The Cluster Template Class

The `ClusterTemplate` class inherits from *Resource*.

class openstack.container_infrastructure_management.v1.cluster_template.ClusterTemplate(*_syn con- nec- tion= **at trs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'clustertemplates'

Plural form of key for resource.

base_path = '/clustertemplates'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_patch = True

Allow patch operation for this resource.

commit_method = 'PATCH'

Method for committing a resource (PUT, PATCH, POST)

commit_jsonpatch = True

Whether commit uses JSON patch format.

apiserver_port

The exposed port of COE API server.

cluster_distro

Display the attribute os_distro defined as appropriate metadata in image for the bay/cluster driver.

coe

Specify the Container Orchestration Engine to use. Supported COEs include kubernetes, swarm, mesos.

created_at

The date and time when the resource was created.

docker_storage_driver

The name of a driver to manage the storage for the images and the containers writable layer.

docker_volume_size

The size in GB for the local storage on each server for the Docker daemon to cache the images and host the containers.

dns_nameserver

The DNS nameserver for the servers and containers in the bay/cluster to use.

external_network_id

The name or network ID of a Neutron network to provide connectivity to the external internet for the bay/cluster.

fixed_network

The name or network ID of a Neutron network to provide connectivity to the internal network for the bay/cluster.

fixed_subnet

Fixed subnet that are using to allocate network address for nodes in bay/cluster.

flavor_id

The nova flavor ID or name for booting the node servers.

http_proxy

The IP address for a proxy to use when direct http access from the servers to sites on the external internet is blocked. This may happen in certain countries or enterprises, and the proxy allows the servers and containers to access these sites. The format is a URL including a port number. The default is None.

https_proxy

The IP address for a proxy to use when direct https access from the servers to sites on the external internet is blocked.

image_id

The name or UUID of the base image in Glance to boot the servers for the bay/cluster.

insecure_registry

The URL pointing to userss own private insecure docker registry to deploy and run docker containers.

is_floating_ip_enabled

Whether enable or not using the floating IP of cloud provider.

is_hidden

Indicates whether the ClusterTemplate is hidden or not.

is_master_lb_enabled

this option can be set to false to create a bay/cluster without the load balancer.

is_tls_disabled

Specifying this parameter will disable TLS so that users can access the COE endpoints without a certificate.

is_public

Setting this flag makes the baymodel/cluster template public and accessible by other users.

is_registry_enabled

This option provides an alternative registry based on the Registry V2

keypair_id

The name of the SSH keypair to configure in the bay/cluster servers for ssh access.

labels

Arbitrary labels. The accepted keys and valid values are defined in the bay/cluster drivers. They are used as a way to pass additional parameters that are specific to a bay/cluster driver.

master_flavor_id

The flavor of the master node for this baymodel/cluster template.

network_driver

The name of a network driver for providing the networks for the containers.

no_proxy

When a proxy server is used, some sites should not go through the proxy and should be accessed normally.

server_type

The servers in the bay/cluster can be vm or baremetal.

updated_at

The date and time when the resource was updated.

uuid

The UUID of the cluster template.

volume_driver

The name of a volume driver for managing the persistent storage for the containers.

openstack.container_infrastructure_management.v1.service

The Service Class

The `Service` class inherits from *Resource*.

class openstack.container_infrastructure_management.v1.service.**Service**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'mservices'

Plural form of key for resource.

base_path = '/mservices'

The base part of the URI for this resource.

allow_list = True

Allow list operation for this resource.

binary

The name of the binary form of the Magnum service.

created_at

The date and time when the resource was created.

disabled_reason

The disable reason of the service, null if the service is enabled or disabled without reason provided.

host

The host for the service.

report_count

The total number of report.

state

The current state of Magnum services.

updated_at

The date and time when the resource was updated.

Database Resources

openstack.database.v1.database

The Database Class

The `Database` class inherits from *Resource*.

class openstack.database.v1.database.**Database**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'database'

Singular form of key for resource.

resources_key = 'databases'

Plural form of key for resource.

base_path = '/instances/%(instance_id)s/databases'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

character_set

Set of symbols and encodings. The default character set is `utf8`.

collate

Set of rules for comparing characters in a character set. The default value for collate is `utf8_general_ci`.

instance_id

The ID of the instance

name

The name of the database

openstack.database.v1.flavor

The Flavor Class

The `Flavor` class inherits from *Resource*.

class openstack.database.v1.flavor.**Flavor**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'flavor'

Singular form of key for resource.

resources_key = 'flavors'

Plural form of key for resource.

base_path = '/flavors'

The base part of the URI for this resource.

allow_list = True

Allow list operation for this resource.

allow_fetch = True

Allow get operation for this resource.

links

Links associated with the flavor

name

The name of the flavor

ram

The size in MB of RAM the flavor has

openstack.database.v1.instance

The Instance Class

The `Instance` class inherits from *Resource*.

class openstack.database.v1.instance.**Instance**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'instance'

Singular form of key for resource.

resources_key = 'instances'

Plural form of key for resource.

base_path = '/instances'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

flavor
The flavor of the instance

links
Links associated with the instance

name
The name of the instance

status
The status of the instance

volume
The size of the volume

datastore
A dictionary of datastore details, often including type and version keys

id
The ID of this instance

region
The region this instance resides in

hostname
The name of the host

created_at
The timestamp when this instance was created

updated_at
The timestamp when this instance was updated

enable_root_user(*session*)
Enable login for the root user.

This operation enables login from any host for the root user and provides the user with a generated root password.

Parameters
session (Adapter) The session to use for making this request.

Returns
A dictionary with keys name and password specifying the login credentials.

is_root_enabled(*session*)
Determine if root is enabled on an instance.

Determine if root is enabled on this particular instance.

Parameters
session (Adapter) The session to use for making this request.

Returns
True if root user is enabled for a specified database instance or False otherwise.

restart(*session*)

Restart the database instance

Returns
None

resize(*session*, *flavor_reference*)

Resize the database instance

Returns
None

resize_volume(*session*, *volume_size*)

Resize the volume attached to the instance

Returns
None

openstack.database.v1.user

The User Class

The `User` class inherits from *Resource*.

class openstack.database.v1.user.**User**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'user'

Singular form of key for resource.

resources_key = 'users'

Plural form of key for resource.

base_path = '/instances/%(instance_id)s/users'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

databases

Databases the user has access to

name

The name of the user

password

The password of the user

DNS Resources

openstack.dns.v2.zone

The Zone Class

The DNS class inherits from *Resource*.

class openstack.dns.v2.zone.**Zone**(*_synchronized=False*, *connection=None*, ***attrs*)

DNS ZONE Resource

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'zones'

Plural form of key for resource.

base_path = '/zones'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

commit_method = 'PATCH'

Method for committing a resource (PUT, PATCH, POST)

action

Properties current action in progress on the resource

attributes

Attributes Key:Value pairs of information about this zone, and the pool the user would like to place the zone in. This information can be used by the scheduler to place zones on the correct pool.

created_at

Timestamp when the zone was created

description

Zone description *Type: str*

email

The administrator email of this zone *Type: str*

links

Links contains a *self* pertaining to this zone or a *next* pertaining to next page

masters

The master list for slaver server to fetch DNS

name

Zone name

pool_id

The pool which manages the zone, assigned by system

project_id

The project id which the zone belongs to

serial

Serial number in the SOA record set in the zone, which identifies the change on the primary DNS server *Type: int*

status

Zone status Valid values include *PENDING_CREATE*, *ACTIVE*, *PENDING_DELETE*, *ERROR*

ttl

SOA TTL time, unit is seconds, default 300, TTL range 300-2147483647 *Type: int*

type

Zone type, Valid values include *PRIMARY*, *SECONDARY Type: str*

updated_at

Timestamp when the zone was last updated

is_shared

Whether the zone is shared with other projects *Type: bool*

delete_shares

If true, delete any existing zone shares along with the zone

openstack.dns.v2.zone_transfer

The ZoneTransferRequest Class

The DNS class inherits from *Resource*.

class openstack.dns.v2.zone_transfer.**ZoneTransferRequest**(*_synchronized=False*, *connection=None*, ***attrs*)

DNS Zone Transfer Request Resource

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

base_path = '/zones/tasks/transfer_requests'

The base part of the URI for this resource.

resources_key = 'transfer_requests'

Plural form of key for resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_commit = True

Allow update operation for this resource.

description

Description

target_project_id

A project ID that the request will be limited to. No other project will be allowed to accept this request.

zone_name

Name for the zone that is being exported

The ZoneTransferAccept Class

The DNS class inherits from *Resource*.

class openstack.dns.v2.zone_transfer.**ZoneTransferAccept**(*_synchronized=False*, *connection=None*, ***attrs*)

DNS Zone Transfer Accept Resource

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

base_path = '/zones/tasks/transfer_accepts'

The base part of the URI for this resource.

resources_key = 'transfer_accepts'

Plural form of key for resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_list = True

Allow list operation for this resource.

zone_transfer_request_id

Name for the zone that is being exported

openstack.dns.v2.zone_export

The ZoneExport Class

The DNS class inherits from *Resource*.

class openstack.dns.v2.zone_export.**ZoneExport**(*_synchronized=False*, *connection=None*, ***attrs*)

DNS Zone Exports Resource

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be

used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = ''

Singular form of key for resource.

resources_key = 'exports'

Plural form of key for resource.

base_path = '/zones/tasks/export'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

created_at

Properties Timestamp when the zone was created

links

Links contains a *self* pertaining to this zone or a *next* pertaining to next page

message

Message

metadata

Returns the total_count of resources matching this filter

project_id

The project id which the zone belongs to

status

Current status of the zone export

updated_at

Timestamp when the zone was last updated

version

Version of the resource

zone_id

ID for the zone that was created by this export

create(*session*, *prepend_key=True*, *base_path=None*, ***kwargs*)

Create a remote resource based on this instance.

Parameters

- **session** (`Adapter`) The session to use for making this request.

- **prepend_key** A boolean indicating whether the resource_key should be prepended in a resource creation request. Default to True.
- **base_path** (*str*) Base part of the URI for creating resources, if different from *base_path*.

Returns
This `Resource` instance.

Raises
MethodNotSupported if `Resource.allow_create` is not set to `True`.

openstack.dns.v2.zone_import

The ZoneImport Class

The `DNS` class inherits from *Resource*.

class `openstack.dns.v2.zone_import.`**ZoneImport**(*_synchronized=False*, *connection=None*, ***attrs*)

DNS Zone Import Resource

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (`openstack.connection.Connection`) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = ''

Singular form of key for resource.

resources_key = 'imports'

Plural form of key for resource.

base_path = '/zones/tasks/import'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

created_at

Properties Timestamp when the zone was created

links
Links contains a *self* pertaining to this zone or a *next* pertaining to next page

message
Message

metadata
Returns the total_count of resources matching this filter

project_id
The project id which the zone belongs to

status
Current status of the zone import

updated_at
Timestamp when the zone was last updated

version
Version of the resource

zone_id
ID for the zone that was created by this import

create(*session*, *prepend_key=True*, *base_path=None*, ***kwargs*)
Create a remote resource based on this instance.

Parameters

- **session** (Adapter) The session to use for making this request.
- **prepend_key** A boolean indicating whether the resource_key should be prepended in a resource creation request. Default to True.
- **base_path** (*str*) Base part of the URI for creating resources, if different from *base_path*.

Returns
This Resource instance.

Raises
MethodNotSupported if Resource.allow_create is not set to True.

openstack.dns.v2.zone_share

The ZoneShare Class

The DNS class inherits from *Resource*.

class openstack.dns.v2.zone_share.**ZoneShare**(*_synchronized=False*, *connection=None*, ***attrs*)

DNS ZONE Share Resource

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.

- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'shared_zones'

Plural form of key for resource.

base_path = '/zones/%(zone_id)s/shares'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_list = True

Allow list operation for this resource.

zone_id

The ID of the zone being shared.

created_at

Timestamp when the share was created.

updated_at

Timestamp when the member was last updated.

project_id

The project ID that owns the share.

target_project_id

The target project ID that the zone is shared with.

openstack.dns.v2.floating_ip

The FloatingIP Class

The DNS class inherits from *Resource*.

class openstack.dns.v2.floating_ip.**FloatingIP**(*_synchronized=False*, *connection=None*, ***attrs*)

DNS Floating IP Resource

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be

used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'floatingips'

Plural form of key for resource.

base_path = '/reverse/floatingips'

The base part of the URI for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_list = True

Allow list operation for this resource.

commit_method = 'PATCH'

Method for committing a resource (PUT, PATCH, POST)

action

Properties current action in progress on the resource

address

The floatingip address for this PTR record

description

Description for this PTR record

ptrdname

Domain name for this PTR record

status

status of the resource

ttl

Time to live for this PTR record

openstack.dns.v2.recordset

The Recordset Class

The DNS class inherits from *Resource*.

class openstack.dns.v2.recordset.**Recordset**(*_synchronized=False*, *connection=None*, ***attrs*)

DNS Recordset Resource

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.

- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'recordsets'

Plural form of key for resource.

base_path = '/zones/%(zone_id)s/recordsets'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

action

Properties current action in progress on the resource

created_at

Timestamp when the zone was created

description

Recordset description

links

Links contains a *self* pertaining to this zone or a *next* pertaining to next page

name

DNS Name of the recordset

project_id

ID of the project which the recordset belongs to

records

DNS record value list

status

Recordset status Valid values include: *PENDING_CREATE*, *ACTIVE*,'PENDING_DELETE', *ERROR*

ttl

Time to live, default 300, available value 300-2147483647 (seconds)

type

DNS type of the recordset Valid values include *A*, *AAAA*, *MX*, *CNAME*, *TXT*, *NS*, *SSHFP*, *SPF*, *SRV*, *PTR*

updated_at

Timestamp when the zone was last updated

zone_id

The id of the Zone which this recordset belongs to

zone_name

The name of the Zone which this recordset belongs to

openstack.dns.v2.limit

The Limit Class

The `Limit` class inherits from *Resource*.

class openstack.dns.v2.limit.**Limit**(*_synchronized=False*, *connection=None*, ***attrs*)

DNS Limit Resource

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'limit'

Singular form of key for resource.

base_path = '/limits'

The base part of the URI for this resource.

allow_list = True

Allow list operation for this resource.

max_page_limit

Properties The max amount of items allowed per page

max_recordset_name_length

The max length of a recordset name

max_recordset_records

The max amount of records contained in a recordset

max_zone_name_length

The max length of a zone name

max_zone_records

The max amount of records in a zone

max_zone_recordsets

The max amount of recordsets per zone

max_zones

The max amount of zones for this project

min_ttl

The lowest ttl allowed on this system

openstack.dns.v2.service_status

The ServiceStatus Class

The `ServiceStatus` class inherits from *Resource*.

class openstack.dns.v2.service_status.**ServiceStatus**(*_synchronized=False*, *connection=None*, ***attrs*)

Designate Service Statuses

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'service_statuses'

Plural form of key for resource.

base_path = '/service_statuses'

The base part of the URI for this resource.

allow_create = False

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = False

Allow update operation for this resource.

allow_delete = False

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

capabilities

Capabilities for the service

created_at

Timestamp when the resource was created

heartbeated_at

Timestamp when the last heartbeat was received

hostname

Hostname of the host with the service instance *Type: str*

links

Links contains a *self* pertaining to this service status or a *next* pertaining to next page

service_name

The name of the Designate service instance *Type: str*

stats

Statistics for the service

status

The status of the resource *Type: enum*

updated_at

Timestamp when the resource was last updated

Identity Resources

Identity v2 Resources

openstack.identity.v2.extension

The Extension Class

The `Extension` class inherits from *Resource*.

class openstack.identity.v2.extension.**Extension**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'extension'

Singular form of key for resource.

resources_key = 'extensions'

Plural form of key for resource.

base_path = '/extensions'

The base part of the URI for this resource.

allow_list = True

Allow list operation for this resource.

allow_fetch = True

Allow get operation for this resource.

alias

A unique identifier, which will be used for accessing the extension through a dedicated url `/extensions/*alias*`. The extension alias uniquely identifies an extension and is prefixed by a vendor identifier. *Type: string*

description

A description of the extension. *Type: string*

links

Links to the documentation in various format. *Type: string*

name

The name of the extension. *Type: string*

namespace

The second unique identifier of the extension after the alias. It is usually a URL which will be used. Example: http://docs.openstack.org/identity/api/ext/s3tokens/v1.0 *Type: string*

updated_at

The last time the extension has been modified (update date).

classmethod list(*session*, *paginated=False*, *base_path=None*, ***params*)

This method is a generator which yields resource objects.

This resource object list generator handles pagination and takes query params for response filtering.

Parameters

- **session** (`Adapter`) The session to use for making this request.
- **paginated** (*bool*) `True` if a GET to this resource returns a paginated series of responses, or `False` if a GET returns only one page of data. **When paginated is False only one page of data will be returned regardless of the APIs support of pagination.**
- **base_path** (*str*) Base part of the URI for listing resources, if different from *base_path*.
- **allow_unknown_params** (*bool*) `True` to accept, but discard unknown query parameters. This allows getting list of filters and passing everything known to the server. `False` will result in validation exception when unknown query parameters are passed.
- **microversion** (*str*) API version to override the negotiated one.
- **headers** (*dict*) Additional headers to inject into the HTTP request.
- **params** (*dict*) These keyword arguments are passed through the `_transpose()` method to find if any of them match expected query parameters to be sent in the *params* argument to `get()`. They are additionally checked against the *base_path* format string to see if any path fragments

need to be filled in by the contents of this argument. Parameters supported as filters by the server side are passed in the API call, remaining parameters are applied as filters to the retrieved results.

Returns
A generator of `Resource` objects.

Raises
MethodNotSupported if `Resource.allow_list` is not set to `True`.

Raises
InvalidResourceQuery if query contains invalid params.

openstack.identity.v2.role

The Role Class

The `Role` class inherits from *Resource*.

class `openstack.identity.v2.role.`**Role**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (`openstack.connection.Connection`) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'role'

Singular form of key for resource.

resources_key = 'roles'

Plural form of key for resource.

base_path = '/OS-KSADM/roles'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

description

The description of the role. *Type: string*

is_enabled

Setting this attribute to `False` prevents this role from being available in the role list. *Type: bool*

name

Unique role name. *Type: string*

openstack.identity.v2.tenant

The Tenant Class

The `Tenant` class inherits from *Resource*.

class openstack.identity.v2.tenant.**Tenant**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'tenant'

Singular form of key for resource.

resources_key = 'tenants'

Plural form of key for resource.

base_path = '/tenants'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

description

The description of the tenant. *Type: string*

is_enabled

Setting this attribute to `False` prevents users from authorizing against this tenant. Additionally, all pre-existing tokens authorized for the tenant are immediately invalidated. Re-enabling a tenant does not re-enable pre-existing tokens. *Type: bool*

name

Unique tenant name. *Type: string*

openstack.identity.v2.user

The User Class

The `User` class inherits from *Resource*.

class `openstack.identity.v2.user.`**User**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (`openstack.connection.Connection`) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'user'

Singular form of key for resource.

resources_key = 'users'

Plural form of key for resource.

base_path = '/users'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

email

The email of this user. *Type: string*

is_enabled

Setting this value to `False` prevents the user from authenticating or receiving authorization. Additionally, all pre-existing tokens held by the user are immediately invalidated. Re-enabling a user does not re-enable pre-existing tokens. *Type: bool*

name

The name of this user. *Type: string*

Identity v3 Resources

openstack.identity.v3.application_credential

The ApplicationCredential Class

The `ApplicationCredential` class inherits from *Resource*.

class openstack.identity.v3.application_credential.**ApplicationCredential**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'application_credential'

Singular form of key for resource.

resources_key = 'application_credentials'

Plural form of key for resource.

base_path = '/users/%(user_id)s/application_credentials'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

user_id

User ID using application credential. *Type: string*

user

User object using application credential. *Type: string*

links

The links for the application credential resource.

name

name of the user. *Type: string*

description

description of application credentials purpose. *Type: string*

expires_at

expire time of application credential. *Type: string*

roles

roles of the user. *Type: list*

unrestricted

restricts the application credential. *Type: boolean*

project_id

ID of project. *Type: string*

access_rules

access rules for application credential. *Type: list*

openstack.identity.v3.credential

The Credential Class

The `Credential` class inherits from *Resource*.

class openstack.identity.v3.credential.**Credential**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'credential'

Singular form of key for resource.

resources_key = 'credentials'

Plural form of key for resource.

base_path = '/credentials'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

commit_method = 'PATCH'

Method for committing a resource (PUT, PATCH, POST)

blob

Arbitrary blob of the credential data, to be parsed according to the `type`. *Type: string*

project_id

References a project ID which limits the scope the credential applies to. This attribute is **mandatory** if the credential type is `ec2`. *Type: string*

type

Representing the credential type, such as `ec2` or `cert`. A specific implementation may determine the list of supported types. *Type: string*

user_id

References the user ID which owns the credential. *Type: string*

openstack.identity.v3.domain

The Domain Class

The `Domain` class inherits from *Resource*.

class openstack.identity.v3.domain.**Domain**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'domain'

Singular form of key for resource.

resources_key = 'domains'

Plural form of key for resource.

base_path = '/domains'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

commit_method = 'PATCH'

Method for committing a resource (PUT, PATCH, POST)

description

The description of this domain. *Type: string*

is_enabled

Setting this attribute to `False` prevents users from authorizing against this domain or any projects owned by this domain, and prevents users owned by this domain from authenticating or receiving any other authorization. Additionally, all pre-existing tokens applicable to the above entities are immediately invalidated. Re-enabling a domain does not re-enable pre-existing tokens. *Type: bool*

options

The resource options for the project. Available resource options are immutable.

links

The links related to the domain resource.

assign_role_to_user(*session*, *user*, *role*, *inherited*)

Assign role to user on domain

validate_user_has_role(*session*, *user*, *role*, *inherited*)

Validates that a user has a role on a domain

unassign_role_from_user(*session*, *user*, *role*, *inherited*)

Unassigns a role from a user on a domain

assign_role_to_group(*session*, *group*, *role*, *inherited*)

Assign role to group on domain

validate_group_has_role(*session*, *group*, *role*, *inherited*)

Validates that a group has a role on a domain

unassign_role_from_group(*session*, *group*, *role*, *inherited*)

Unassigns a role from a group on a domain

openstack.identity.v3.domain_config

The Domain Class

The DomainConfig class inherits from *Resource*.

class openstack.identity.v3.domain_config.**DomainConfig**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of self._connection in Resource code should protect itself with a check for None.

resource_key = 'config'

Singular form of key for resource.

base_path = '/domains/%(domain_id)s/config'

The base part of the URI for this resource.

requires_id = False

Do calls for this resource require an id

create_requires_id = False

Whether create requires an ID (determined from method if None).

commit_method = 'PATCH'

Method for committing a resource (PUT, PATCH, POST)

create_method = 'PUT'

Method for creating a resource (POST, PUT)

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

domain_id

The domain ID.

identity

An identity object.

ldap

The config object.

openstack.identity.v3.endpoint

The Endpoint Class

The Endpoint class inherits from *Resource*.

class openstack.identity.v3.endpoint.**Endpoint**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of self._connection in Resource code should protect itself with a check for None.

resource_key = 'endpoint'

Singular form of key for resource.

resources_key = 'endpoints'

Plural form of key for resource.

base_path = '/endpoints'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

commit_method = 'PATCH'

Method for committing a resource (PUT, PATCH, POST)

interface

Describes the interface of the endpoint according to one of the following values:

- ***public*: intended for consumption by end users, generally on a** publicly available network interface
- ***internal*: not intended for consumption by end users, generally on an** unmetered internal network interface
- ***admin*: intended only for consumption by those needing administrative** access to the service, generally on a secure network interface

Type: string

is_enabled

Setting this value to `False` prevents the endpoint from appearing in the service catalog. *Type: bool*

links

The links for the region resource.

region_id

Represents the containing region ID of the service endpoint. *New in v3.2 Type: string*

service_id

References the service ID to which the endpoint belongs. *Type: string*

url

Fully qualified URL of the service endpoint. *Type: string*

openstack.identity.v3.federation_protocol

The FederationProtocol Class

The `FederationProtocol` class inherits from *Resource*.

class openstack.identity.v3.federation_protocol.**FederationProtocol**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'protocol'

Singular form of key for resource.

resources_key = 'protocols'

Plural form of key for resource.

base_path = '/OS-FEDERATION/identity_providers/%(idp_id)s/protocols'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

create_exclude_id_from_body = True

Whether create should exclude ID in the body of the request.

create_method = 'PUT'

Method for creating a resource (POST, PUT)

commit_method = 'PATCH'

Method for committing a resource (PUT, PATCH, POST)

name

name of the protocol (read only) *Type: string*

openstack.identity.v3.group

The Group Class

The Group class inherits from *Resource*.

class openstack.identity.v3.group.**Group**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'group'

Singular form of key for resource.

resources_key = 'groups'

Plural form of key for resource.

base_path = '/groups'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

commit_method = 'PATCH'

Method for committing a resource (PUT, PATCH, POST)

description

The description of this group. *Type: string*

domain_id

References the domain ID which owns the group; if a domain ID is not specified by the client, the Identity service implementation will default it to the domain ID to which the clients token is scoped. *Type: string*

name

Unique group name, within the owning domain. *Type: string*

add_user(*session*, *user*)

Add user to the group

remove_user(*session*, *user*)

Remove user from the group

check_user(*session*, *user*)

Check whether user belongs to group

openstack.identity.v3.identity_provider

The IdentityProvider Class

The `IdentityProvider` class inherits from *Resource*.

class openstack.identity.v3.identity_provider.**IdentityProvider**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.

- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'identity_provider'

Singular form of key for resource.

resources_key = 'identity_providers'

Plural form of key for resource.

base_path = '/OS-FEDERATION/identity_providers'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

create_method = 'PUT'

Method for creating a resource (POST, PUT)

create_exclude_id_from_body = True

Whether create should exclude ID in the body of the request.

commit_method = 'PATCH'

Method for committing a resource (PUT, PATCH, POST)

description

A description of this identity provider. *Type: string*

is_enabled

If the identity provider is currently enabled. *Type: bool*

remote_ids

Remote IDs associated with the identity provider. *Type: list*

name

The identifier of the identity provider (read only). *Type: string*

openstack.identity.v3.limit

The Limit Class

The `Limit` class inherits from *Resource*.

class openstack.identity.v3.limit.**Limit**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'limit'

Singular form of key for resource.

resources_key = 'limits'

Plural form of key for resource.

base_path = '/limits'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

commit_method = 'PATCH'

Method for committing a resource (PUT, PATCH, POST)

commit_jsonpatch = True

Whether commit uses JSON patch format.

description

User-facing description of the registered_limit. *Type: string*

links

The links for the registered_limit resource.

service_id

ID of service. *Type: string*

region_id

ID of region, if any. *Type: string*

resource_name

The resource name. *Type: string*

resource_limit

The resource limit value. *Type: int*

project_id

ID of project. *Type: string*

create(*session*, *prepend_key=True*, *base_path=None*, ***, *resource_request_key=None*, *resource_response_key='limits'*, *microversion=None*, ***params*)

Create a remote resource based on this instance.

Parameters

- **session** (Adapter) The session to use for making this request.
- **prepend_key** A boolean indicating whether the resource_key should be prepended in a resource creation request. Default to True.
- **base_path** (*str*) Base part of the URI for creating resources, if different from *base_path*.
- **resource_request_key** (*str*) Overrides the usage of self.resource_key when prepending a key to the request body. Ignored if *prepend_key* is false.
- **resource_response_key** (*str*) Overrides the usage of self.resource_key when processing response bodies. Ignored if *prepend_key* is false.
- **microversion** (*str*) API version to override the negotiated one.
- **params** (*dict*) Additional params to pass.

Returns

This Resource instance.

Raises

MethodNotSupported if Resource.allow_create is not set to True.

openstack.identity.v3.mapping

The Mapping Class

The Mapping class inherits from *Resource*.

class openstack.identity.v3.mapping.**Mapping**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of self._connection in Resource code should protect itself with a check for None.

resource_key = 'mapping'

Singular form of key for resource.

resources_key = 'mappings'

Plural form of key for resource.

base_path = '/OS-FEDERATION/mappings'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

create_method = 'PUT'

Method for creating a resource (POST, PUT)

commit_method = 'PATCH'

Method for committing a resource (PUT, PATCH, POST)

rules

The rules of this mapping. *Type: list*

name

The identifier of the mapping. *Type: string*

openstack.identity.v3.policy

The Policy Class

The `Policy` class inherits from *Resource*.

class openstack.identity.v3.policy.**Policy**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'policy'

Singular form of key for resource.

resources_key = 'policies'

Plural form of key for resource.

base_path = '/policies'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

commit_method = 'PATCH'

Method for committing a resource (PUT, PATCH, POST)

blob

The policy rule set itself, as a serialized blob. *Type: string*

links

The links for the policy resource.

project_id

The ID for the project.

type

The MIME Media Type of the serialized policy blob. *Type: string*

user_id

The ID of the user who owns the policy

openstack.identity.v3.project

The Project Class

The `Project` class inherits from *Resource*.

class openstack.identity.v3.project.**Project**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be

used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'project'

Singular form of key for resource.

resources_key = 'projects'

Plural form of key for resource.

base_path = '/projects'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

commit_method = 'PATCH'

Method for committing a resource (PUT, PATCH, POST)

description

The description of the project. *Type: string*

domain_id

References the domain ID which owns the project; if a domain ID is not specified by the client, the Identity service implementation will default it to the domain ID to which the clients token is scoped. *Type: string*

is_domain

Indicates whether the project also acts as a domain. If set to True, the project acts as both a project and a domain. Default is False. New in version 3.6

is_enabled

Setting this attribute to `False` prevents users from authorizing against this project. Additionally, all pre-existing tokens authorized for the project are immediately invalidated. Re-enabling a project does not re-enable pre-existing tokens. *Type: bool*

options

The resource options for the project. Available resource options are immutable.

parent_id

The ID of the parent of the project. New in version 3.4

links

The links related to the project resource.

assign_role_to_user(*session*, *user*, *role*, *inherited*)

Assign role to user on project

validate_user_has_role(*session*, *user*, *role*, *inherited*)

Validates that a user has a role on a project

unassign_role_from_user(*session*, *user*, *role*, *inherited*)

Unassigns a role from a user on a project

assign_role_to_group(*session*, *group*, *role*, *inherited*)

Assign role to group on project

validate_group_has_role(*session*, *group*, *role*, *inherited*)

Validates that a group has a role on a project

unassign_role_from_group(*session*, *group*, *role*, *inherited*)

Unassigns a role from a group on a project

associate_endpoint(*session*, *endpoint_id*)

Associate endpoint with project.

Parameters

- **session** The session to use for making this request.
- **endpoint_id** The ID of an endpoint.

Returns

None

disassociate_endpoint(*session*, *endpoint_id*)

Disassociate endpoint from project.

Parameters

- **session** The session to use for making this request.
- **endpoint_id** The ID of an endpoint.

Returns

None

openstack.identity.v3.region

The Region Class

The `Region` class inherits from *Resource*.

class openstack.identity.v3.region.**Region**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.

- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'region'

Singular form of key for resource.

resources_key = 'regions'

Plural form of key for resource.

base_path = '/regions'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

commit_method = 'PATCH'

Method for committing a resource (PUT, PATCH, POST)

description

User-facing description of the region. *Type: string*

links

The links for the region resource.

parent_region_id

ID of parent region, if any. *Type: string*

openstack.identity.v3.registered_limit

The RegisteredLimit Class

The `RegisteredLimit` class inherits from *Resource*.

class openstack.identity.v3.registered_limit.**RegisteredLimit**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.

- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'registered_limit'

Singular form of key for resource.

resources_key = 'registered_limits'

Plural form of key for resource.

base_path = '/registered_limits'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

commit_method = 'PATCH'

Method for committing a resource (PUT, PATCH, POST)

commit_jsonpatch = True

Whether commit uses JSON patch format.

description

User-facing description of the registered_limit. *Type: string*

links

The links for the registered_limit resource.

service_id

ID of service. *Type: string*

region_id

ID of region, if any. *Type: string*

resource_name

The resource name. *Type: string*

default_limit

The default limit value. *Type: int*

create(*session*, *prepend_key=True*, *base_path=None*, ***, *resource_request_key=None*, *resource_response_key='registered_limits'*, *microversion=None*, ***params*)

Create a remote resource based on this instance.

Parameters

- **session** (Adapter) The session to use for making this request.
- **prepend_key** A boolean indicating whether the resource_key should be prepended in a resource creation request. Default to True.
- **base_path** (*str*) Base part of the URI for creating resources, if different from *base_path*.
- **resource_request_key** (*str*) Overrides the usage of self.resource_key when prepending a key to the request body. Ignored if *prepend_key* is false.
- **resource_response_key** (*str*) Overrides the usage of self.resource_key when processing response bodies. Ignored if *prepend_key* is false.
- **microversion** (*str*) API version to override the negotiated one.
- **params** (*dict*) Additional params to pass.

Returns

This Resource instance.

Raises

MethodNotSupported if Resource.allow_create is not set to True.

openstack.identity.v3.role

The Role Class

The Role class inherits from *Resource*.

class openstack.identity.v3.role.**Role**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of self._connection in Resource code should protect itself with a check for None.

resource_key = 'role'

Singular form of key for resource.

resources_key = 'roles'

Plural form of key for resource.

base_path = '/roles'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

commit_method = 'PATCH'

Method for committing a resource (PUT, PATCH, POST)

name

Unique role name, within the owning domain. *Type: string*

description

User-facing description of the role. *Type: string*

domain_id

References the domain ID which owns the role. *Type: string*

options

The resource options for the role. Available resource options are immutable.

links

The links for the service resource.

openstack.identity.v3.role_assignment

The RoleAssignment Class

The `RoleAssignment` class inherits from *Resource*.

class openstack.identity.v3.role_assignment.**RoleAssignment**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'role_assignment'

Singular form of key for resource.

resources_key = 'role_assignments'

Plural form of key for resource.

base_path = '/role_assignments'

The base part of the URI for this resource.

allow_list = True

Allow list operation for this resource.

links

The links for the service resource.

role

The role (dictionary contains only id) *Type: dict*

scope

The scope (either domain or project; dictionary contains only id) *Type: dict*

user

The user (dictionary contains only id) *Type: dict*

group

The group (dictionary contains only id) *Type: dict*

openstack.identity.v3.role_domain_group_assignment

The RoleDomainGroupAssignment Class

The `RoleDomainGroupAssignment` class inherits from *Resource*.

class openstack.identity.v3.role_domain_group_assignment.**RoleDomainGroupAssignment**(*_synchroni*
con-
nec-
tion=None
***at-*
trs)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (*openstack.connection.Connection*) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'role'

Singular form of key for resource.

resources_key = 'roles'

Plural form of key for resource.

base_path = '/domains/%(domain_id)s/groups/%(group_id)s/roles'

The base part of the URI for this resource.

allow_list = True

Allow list operation for this resource.

name

name of the role *Type: string*

links

The links for the service resource.

domain_id

The ID of the domain to list assignment from. *Type: string*

group_id

The ID of the group to list assignment from. *Type: string*

openstack.identity.v3.role_domain_user_assignment

The RoleDomainUserAssignment Class

The `RoleDomainUserAssignment` class inherits from *Resource*.

class openstack.identity.v3.role_domain_user_assignment.**RoleDomainUserAssignment**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'role'

Singular form of key for resource.

resources_key = 'roles'

Plural form of key for resource.

base_path = '/domains/%(domain_id)s/users/%(user_id)s/roles'

The base part of the URI for this resource.

allow_list = True

Allow list operation for this resource.

name

name of the role *Type: string*

links

The links for the service resource.

domain_id

The ID of the domain to list assignment from. *Type: string*

user_id

The ID of the user to list assignment from. *Type: string*

openstack.identity.v3.role_project_group_assignment

The RoleProjectGroupAssignment Class

The `RoleProjectGroupAssignment` class inherits from *Resource*.

class openstack.identity.v3.role_project_group_assignment.**RoleProjectGroupAssignment**(*_synchr con- nec- tion=No **at- trs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'role'

Singular form of key for resource.

resources_key = 'roles'

Plural form of key for resource.

base_path = '/projects/%(project_id)s/groups/%(group_id)s/roles'

The base part of the URI for this resource.

allow_list = True

Allow list operation for this resource.

name

name of the role *Type: string*

links

The links for the service resource.

project_id

The ID of the project to list assignment from. *Type: string*

group_id

The ID of the group to list assignment from. *Type: string*

openstack.identity.v3.role_project_user_assignment

The RoleProjectUserAssignment Class

The `RoleProjectUserAssignment` class inherits from *Resource*.

class openstack.identity.v3.role_project_user_assignment.**RoleProjectUserAssignment**(*_synchroni*
con-
nec-
tion=None
***at-*
trs)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'role'

Singular form of key for resource.

resources_key = 'roles'

Plural form of key for resource.

base_path = '/projects/%(project_id)s/users/%(user_id)s/roles'

The base part of the URI for this resource.

allow_list = True

Allow list operation for this resource.

name

name of the role *Type: string*

links

The links for the service resource.

project_id

The ID of the project to list assignment from. *Type: string*

user_id

The ID of the user to list assignment from. *Type: string*

openstack.identity.v3.role_system_group_assignment

The RoleSystemGroupAssignment Class

The `RoleSystemGroupAssignment` class inherits from *Resource*.

class openstack.identity.v3.role_system_group_assignment.**RoleSystemGroupAssignment**(*_synchroni*
con-
nec-
tion=None
***at-*
trs)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'role'

Singular form of key for resource.

resources_key = 'roles'

Plural form of key for resource.

base_path = '/system/groups/%(group_id)s/roles'

The base part of the URI for this resource.

allow_list = True

Allow list operation for this resource.

group_id

The ID of the group to list assignment from. *Type: string*

system_id

The name of the system to list assignment from. *Type: string*

openstack.identity.v3.role_system_user_assignment

The RoleSystemUserAssignment Class

The `RoleSystemUserAssignment` class inherits from *Resource*.

class openstack.identity.v3.role_system_user_assignment.**RoleSystemUserAssignment**(*_synchronized*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'role'

Singular form of key for resource.

resources_key = 'roles'

Plural form of key for resource.

base_path = '/system/users/%(user_id)s/roles'

The base part of the URI for this resource.

allow_list = True

Allow list operation for this resource.

system_id

The name of the system to list assignment from. *Type: string*

user_id

The ID of the user to list assignment from. *Type: string*

openstack.identity.v3.service

The Service Class

The `Service` class inherits from *Resource*.

class openstack.identity.v3.service.**Service**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'service'

Singular form of key for resource.

resources_key = 'services'

Plural form of key for resource.

base_path = '/services'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

commit_method = 'PATCH'

Method for committing a resource (PUT, PATCH, POST)

description

User-facing description of the service. *Type: string*

is_enabled

Setting this value to `False` prevents the service and its endpoints from appearing in the service catalog. *Type: bool*

links

The links for the service resource.

name

User-facing name of the service. *Type: string*

type

Describes the API implemented by the service. The following values are recognized within the OpenStack ecosystem: `compute`, `image`, `ec2`, `identity`, `volume`, `network`. To support non-core and future projects, the value should not be validated against this list. *Type: string*

openstack.identity.v3.system

The System Class

The `System` class inherits from *Resource*.

class openstack.identity.v3.system.**System**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'system'

Singular form of key for resource.

base_path = '/system'

The base part of the URI for this resource.

assign_role_to_user(*session*, *user*, *role*)

Assign role to user on system

validate_user_has_role(*session*, *user*, *role*)

Validates that a user has a role on a system

unassign_role_from_user(*session*, *user*, *role*)

Unassigns a role from a user on a system

assign_role_to_group(*session*, *group*, *role*)

Assign role to group on system

validate_group_has_role(*session*, *group*, *role*)

Validates that a group has a role on a system

unassign_role_from_group(*session*, *group*, *role*)

Unassigns a role from a group on a system

openstack.identity.v3.trust

The Trust Class

The `Trust` class inherits from *Resource*.

class openstack.identity.v3.trust.**Trust**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'trust'

Singular form of key for resource.

resources_key = 'trusts'

Plural form of key for resource.

base_path = '/OS-TRUST/trusts'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_redelegation

A boolean indicating whether the trust can be issued by the trustee as a regulart trust. Default is `False`.

expires_at

Specifies the expiration time of the trust. A trust may be revoked ahead of expiration. If the value represents a time in the past, the trust is deactivated.

is_impersonation

If `impersonation` is set to true, then the `user` attribute of tokens that are generated based on the trust will represent that of the trustor rather than the trustee, thus allowing the trustee to impersonate the trustor. If `impersonation` is set to `False`, then the tokens `user` attribute will represent that of the trustee. *Type: bool*

links

Links for the trust resource.

project_id

ID of the project upon which the trustor is delegating authorization. *Type: string*

role_links

A role links object that includes next, previous, and self links for roles.

roles

Specifies the subset of the trustors roles on the `project_id` to be granted to the trustee when the token in consumed. The trustor must already be granted these roles in the project referenced by the `project_id` attribute. *Type: list*

redelegated_trust_id

Returned with redelegated trust provides information about the predecessor in the trust chain.

redelegation_count

Redelegation count

remaining_uses

How many times the trust can be used to obtain a token. The value is decreased each time a token is issued through the trust. Once it reaches zero, no further tokens will be isued through the trust.

trustee_user_id

Represents the user ID who is capable of consuming the trust. *Type: string*

trustor_user_id

Represents the user ID who created the trust, and whos authorization is being delegated. *Type: string*

openstack.identity.v3.user

The User Class

The `User` class inherits from *Resource*.

class openstack.identity.v3.user.**User**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.

- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'user'

Singular form of key for resource.

resources_key = 'users'

Plural form of key for resource.

base_path = '/users'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

commit_method = 'PATCH'

Method for committing a resource (PUT, PATCH, POST)

default_project_id

References the users default project ID against which to authorize, if the API user does not explicitly specify one when creating a token. Setting this attribute does not grant any actual authorization on the project, and is merely provided for the users convenience. Therefore, the referenced project does not need to exist within the users domain.

New in version 3.1 If the user does not have authorization to their default project, the default project will be ignored at token creation. *Type: string*

description

The description of this user. *Type: string*

domain_id

References the domain ID which owns the user; if a domain ID is not specified by the client, the Identity service implementation will default it to the domain ID to which the clients token is scoped. *Type: string*

email

The email of this user. *Type: string*

is_enabled

Setting this value to `False` prevents the user from authenticating or receiving authorization. Additionally, all pre-existing tokens held by the user are immediately invalidated. Re-enabling a user does not re-enable pre-existing tokens. *Type: bool*

links

The links for the user resource.

name

Unique user name, within the owning domain. *Type: string*

password

The default form of credential used during authentication. *Type: string*

password_expires_at

The date and time when the password expires. The time zone is UTC. A None value means the password never expires. This is a response object attribute, not valid for requests. *New in version 3.7*

options

A dictionary of users extra options.

Other Resources

openstack.identity.version

The Version Class

The `Version` class inherits from *Resource*.

class openstack.identity.version.**Version**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'version'

Singular form of key for resource.

resources_key = 'versions'

Plural form of key for resource.

base_path = '/'

The base part of the URI for this resource.

allow_list = True

Allow list operation for this resource.

classmethod list(*session*, *paginated=False*, *base_path=None*, ***params*)

This method is a generator which yields resource objects.

This resource object list generator handles pagination and takes query params for response filtering.

Parameters

- **session** (Adapter) The session to use for making this request.
- **paginated** (*bool*) True if a GET to this resource returns a paginated series of responses, or False if a GET returns only one page of data. **When paginated is False only one page of data will be returned regardless of the APIs support of pagination.**
- **base_path** (*str*) Base part of the URI for listing resources, if different from *base_path*.
- **allow_unknown_params** (*bool*) True to accept, but discard unknown query parameters. This allows getting list of filters and passing everything known to the server. False will result in validation exception when unknown query parameters are passed.
- **microversion** (*str*) API version to override the negotiated one.
- **headers** (*dict*) Additional headers to inject into the HTTP request.
- **params** (*dict*) These keyword arguments are passed through the _transpose() method to find if any of them match expected query parameters to be sent in the *params* argument to get(). They are additionally checked against the *base_path* format string to see if any path fragments need to be filled in by the contents of this argument. Parameters supported as filters by the server side are passed in the API call, remaining parameters are applied as filters to the retrieved results.

Returns
A generator of Resource objects.

Raises
MethodNotSupported if Resource.allow_list is not set to True.

Raises
InvalidResourceQuery if query contains invalid params.

Image Resources

Image v1 Resources

openstack.image.v1.image

The Image Class

The Image class inherits from *Resource*.

class openstack.image.v1.image.**Image**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be

used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'image'

Singular form of key for resource.

resources_key = 'images'

Plural form of key for resource.

base_path = '/images'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

checksum

Hash of the image data used. The Image service uses this value for verification.

container_format

The container format refers to whether the VM image is in a file format that also contains metadata about the actual VM. Container formats include OVF and Amazon AMI. In addition, a VM image might not have a container format - instead, the image is just a blob of unstructured data.

copy_from

A URL to copy an image from

created_at

The timestamp when this image was created.

disk_format

Valid values are: aki, ari, ami, raw, iso, vhd, vdi, qcow2, or vmdk. The disk format of a VM image is the format of the underlying disk image. Virtual appliance vendors have different formats for laying out the information contained in a VM disk image.

is_protected

Defines whether the image can be deleted. *Type: bool*

is_public

`True` if this is a public image. *Type: bool*

location

A location for the image identified by a URI

min_disk

The minimum disk size in GB that is required to boot the image.

min_ram

The minimum amount of RAM in MB that is required to boot the image.

name

Name for the image. Note that the name of an image is not unique to a Glance node. The API cannot expect users to know the names of images owned by others.

owner

The ID of the owner, or project, of the image.

owner_id

The ID of the owner, or project, of the image. (backwards compat)

properties

Properties, if any, that are associated with the image.

size

The size of the image data, in bytes.

status

The image status.

updated_at

The timestamp when this image was last updated.

classmethod find(*session*, *name_or_id*, *ignore_missing=True*, ***params*)

Find a resource by its name or id.

Parameters

- **session** (Adapter) The session to use for making this request.
- **name_or_id** This resources identifier, if needed by the request. The default is None.
- **ignore_missing** (*bool*) When set to False *NotFoundException* will be raised when the resource does not exist. When set to True, None will be returned when attempting to find a nonexistent resource.
- **params** (*dict*) Any additional parameters to be passed into underlying methods, such as to *existing()* in order to pass on URI parameters.

Returns

The Resource object matching the given name or id or None if nothing matches.

Raises

openstack.exceptions.DuplicateResource if more than one resource is found for this request.

Raises

openstack.exceptions.NotFoundException if nothing is found and ignore_missing is False.

Image v2 Resources

openstack.image.v2.image

The Image Class

The Image class inherits from *Resource*.

class openstack.image.v2.image.**Image**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of self._connection in Resource code should protect itself with a check for None.

resources_key = 'images'

Plural form of key for resource.

base_path = '/images'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

commit_method = 'PATCH'

Method for committing a resource (PUT, PATCH, POST)

commit_jsonpatch = True

Whether commit uses JSON patch format.

checksum

Hash of the image data used. The Image service uses this value for verification.

container_format

The container format refers to whether the VM image is in a file format that also contains metadata about the actual VM. Container formats include OVF and Amazon AMI. In addition, a VM image might not have a container format - instead, the image is just a blob of unstructured data.

created_at

The date and time when the image was created.

disk_format

Valid values are: aki, ari, ami, raw, iso, vhd, vdi, qcow2, or vmdk. The disk format of a VM image is the format of the underlying disk image. Virtual appliance vendors have different formats for laying out the information contained in a VM disk image.

is_hidden

This field controls whether an image is displayed in the default image-list response

is_protected

Defines whether the image can be deleted. *Type: bool*

hash_algo

The algorithm used to compute a secure hash of the image data for this image

hash_value

The hexdigest of the secure hash of the image data computed using the algorithm whose name is the value of the os_hash_algo property.

min_disk

The minimum disk size in GB that is required to boot the image.

min_ram

The minimum amount of RAM in MB that is required to boot the image.

name

The name of the image.

owner

The ID of the owner, or project, of the image.

owner_id

The ID of the owner, or project, of the image. (backwards compat)

properties

Properties, if any, that are associated with the image.

size

The size of the image data, in bytes.

store

When present, Glance will attempt to store the disk image data in the backing store indicated by the value of the header. When not present, Glance will store the disk image data in the backing store that is marked default. Valid values are: file, s3, rbd, swift, cinder, gridfs, sheepdog, or vsphere.

status

The image status.

updated_at

The date and time when the image was updated.

virtual_size

The virtual size of the image.

visibility

The image visibility.

file

The URL for the virtual machine image file.

locations

A list of URLs to access the image file in external store. This list appears if the show_multiple_locations option is set to true in the Image services configuration file.

direct_url

The URL to access the image file kept in external store. It appears when you set the show_image_direct_url option to true in the Image services configuration file.

url

The URL to access the image file kept in external store.

metadata

The location metadata.

architecture

The CPU architecture that must be supported by the hypervisor.

hypervisor_type

The hypervisor type. Note that qemu is used for both QEMU and KVM hypervisor types.

instance_type_rxtx_factor

Optional property allows created servers to have a different bandwidth cap than that defined in the network they are attached to.

instance_uuid

create this image.

needs_config_drive

Specifies whether the image needs a config drive. *mandatory* or *optional* (default if property is not used).

kernel_id

The ID of an image stored in the Image service that should be used as the kernel when booting an AMI-style image.

os_distro

The common name of the operating system distribution in lowercase

os_version

The operating system version as specified by the distributor.

needs_secure_boot

Secure Boot is a security standard. When the instance starts, Secure Boot first examines software such as firmware and OS by their signature and only allows them to run if the signatures are valid.

os_shutdown_timeout

Time for graceful shutdown

ramdisk_id

The ID of image stored in the Image service that should be used as the ramdisk when booting an AMI-style image.

vm_mode

The virtual machine mode. This represents the host/guest ABI (application binary interface) used for the virtual machine.

hw_cpu_sockets

The preferred number of sockets to expose to the guest.

hw_cpu_cores

The preferred number of cores to expose to the guest.

hw_cpu_threads

The preferred number of threads to expose to the guest.

hw_disk_bus

Specifies the type of disk controller to attach disk devices to. One of scsi, virtio, uml, xen, ide, or usb.

hw_cpu_policy

Used to pin the virtual CPUs (vCPUs) of instances to the hosts physical CPU cores (pCPUs).

hw_cpu_thread_policy

Defines how hardware CPU threads in a simultaneous multithreading-based (SMT) architecture be used.

hw_rng_model

Adds a random-number generator device to the images instances.

hw_machine_type

For libvirt: Enables booting an ARM system using the specified machine type. For Hyper-V: Specifies whether the Hyper-V instance will be a generation 1 or generation 2 VM.

hw_scsi_model

Enables the use of VirtIO SCSI (virtio-scsi) to provide block device access for compute instances; by default, instances use VirtIO Block (virtio-blk).

hw_serial_port_count

Specifies the count of serial ports that should be provided.

hw_video_model

The video image driver used.

hw_video_ram

Maximum RAM for the video image.

hw_watchdog_action

Enables a virtual hardware watchdog device that carries out the specified action if the server hangs.

os_command_line
: The kernel command line to be used by the libvirt driver, instead of the default.

hw_vif_model
: Specifies the model of virtual network interface device to use.

is_hw_vif_multiqueue_enabled
: If true, this enables the virtio-net multiqueue feature. In this case, the driver sets the number of queues equal to the number of guest vCPUs. This makes the network performance scale across a number of vCPUs.

is_hw_boot_menu_enabled
: If true, enables the BIOS bootmenu.

vmware_adaptertype
: The virtual SCSI or IDE controller used by the hypervisor.

vmware_ostype
: A VMware GuestID which describes the operating system installed in the image.

has_auto_disk_config
: If true, the root partition on the disk is automatically resized before the instance boots.

os_type
: The operating system installed on the image.

os_admin_user
: The operating system admin username.

hw_qemu_guest_agent
: A string boolean, which if true, QEMU guest agent will be exposed to the instance.

os_require_quiesce
: If true, require quiesce on snapshot via QEMU guest agent.

schema
: The URL for the schema describing a virtual machine image.

deactivate(*session*)
: Deactivate an image

  Note: Only administrative users can view image locations for deactivated images.

reactivate(*session*)
: Reactivate an image

  Note: The image must exist in order to be reactivated.

upload(*session*, *, *data=None*)
: Upload data into an existing image

  Parameters

  - **session** The session to use for making this request
  - **data** Optional data to be uploaded. If not provided, the *~Image.data* attribute will be used

Returns
The server response

stage(*session*, ***, *data=None*)

Stage binary image data into an existing image

Parameters

- **session** The session to use for making this request
- **data** Optional data to be uploaded. If not provided, the *~Image.data* attribute will be used

Returns
The server response

import_image(*session*, *method='glance-direct'*, ***, *uri=None*, *remote_region=None*, *remote_image_id=None*, *remote_service_interface=None*, *store=None*, *stores=None*, *all_stores=None*, *all_stores_must_succeed=None*)

Import Image via interoperable image import process

classmethod find(*session*, *name_or_id*, *ignore_missing=True*, ***params*)

Find a resource by its name or id.

Parameters

- **session** (Adapter) The session to use for making this request.
- **name_or_id** This resources identifier, if needed by the request. The default is None.
- **ignore_missing** (*bool*) When set to False *NotFoundException* will be raised when the resource does not exist. When set to True, None will be returned when attempting to find a nonexistent resource.
- **list_base_path** (*str*) base_path to be used when need listing resources.
- **microversion** (*str*) API version to override the negotiated one.
- **params** (*dict*) Any additional parameters to be passed into underlying methods, such as to *existing()* in order to pass on URI parameters.

Returns
The Resource object matching the given name or id or None if nothing matches.

Raises
openstack.exceptions.DuplicateResource if more than one resource is found for this request.

Raises
openstack.exceptions.NotFoundException if nothing is found and ignore_missing is False.

openstack.image.v2.member

The Member Class

The `Member` class inherits from *Resource*.

class openstack.image.v2.member.**Member**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'members'

Plural form of key for resource.

base_path = '/images/%(image_id)s/members'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

member_id

The ID of the image member. An image member is a tenant with whom the image is shared.

created_at

The date and time when the member was created.

image_id

Image ID stored through the image API. Typically a UUID.

status

The status of the image.

schema

The URL for schema of the member.

updated_at

The date and time when the member was updated.

openstack.image.v2.metadef_namespace

The MetadefNamespace Class

The `MetadefNamespace` class inherits from *Resource*.

class openstack.image.v2.metadef_namespace.**MetadefNamespace**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'namespaces'

Plural form of key for resource.

base_path = '/metadefs/namespaces'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_delete = True

Allow delete operation for this resource.

delete_all_properties(*session*)

Delete all properties in a namespace.

Parameters
session The session to use for making this request

Returns
The server response

delete_all_objects(*session*)

Delete all objects in a namespace.

Parameters
session The session to use for making this request

Returns
The server response

openstack.image.v2.metadef_object

The MetadefObject Class

The `MetadefObject` class inherits from *Resource*.

`class openstack.image.v2.metadef_object.MetadefObject(_synchronized=False, connection=None, **attrs)`

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'objects'

Plural form of key for resource.

base_path = '/metadefs/namespaces/%(namespace_name)s/objects'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

name

The name of this resource.

openstack.image.v2.metadef_resource_type

The MetadefResourceType Class

The `MetadefResourceType` class inherits from *Resource*.

`class openstack.image.v2.metadef_resource_type.MetadefResourceType(_synchronized=False, connection=None, **attrs)`

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'resource_types'

Plural form of key for resource.

base_path = '/metadefs/resource_types'

The base part of the URI for this resource.

allow_list = True

Allow list operation for this resource.

name

The name of metadata definition resource type

created_at

The date and time when the resource type was created.

updated_at

The date and time when the resource type was updated.

The MetadefResourceTypeAssociation Class

The `MetadefResourceTypeAssociation` class inherits from *Resource*.

class openstack.image.v2.metadef_resource_type.**MetadefResourceTypeAssociation**(*_synchronized=F connection=None, **attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'resource_type_associations'

Plural form of key for resource.

base_path = '/metadefs/namespaces/%(namespace_name)s/resource_types'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

namespace_name

The name of the namespace whose details you want to see.

name

The name of metadata definition resource type

created_at

The date and time when the resource type was created.

updated_at

The date and time when the resource type was updated.

prefix

Prefix for any properties in the namespace that you want to apply to the resource type. If you specify a prefix, you must append a prefix separator, such as the colon (:) character.

properties_target

Some resource types allow more than one key and value pair for each instance. For example, the Image service allows both user and image metadata on volumes. The properties_target parameter enables a namespace target to remove the ambiguity

openstack.image.v2.metadef_property

The MetadefProperty Class

The `MetadefProperty` class inherits from *Resource*.

class openstack.image.v2.metadef_property.**MetadefProperty**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

base_path = '/metadefs/namespaces/%(namespace_name)s/properties'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

namespace_name

An identifier (a name) for the namespace.

name

The name of the property

type

The property type.

title

The title of the property.

description

Detailed description of the property.

operators

A list of operator

default

Default property description.

is_readonly

Indicates whether this is a read-only property.

minimum

Minimum allowed numerical value.

maximum

Maximum allowed numerical value.

enum

Enumerated list of property values.

pattern

A regular expression (ECMA 262) that a string value must match.

min_length

Minimum allowed string length.

max_length

Maximum allowed string length.

items

Schema for the items in an array.

require_unique_items

Indicates whether all values in the array must be distinct.

min_items

Minimum length of an array.

max_items

Maximum length of an array.

allow_additional_items

Describes extra items, if you use tuple typing. If the value of `items` is an array (tuple typing) and the instance is longer than the list of schemas in `items`, the additional items are described by the schema in this property. If this value is `false`, the instance cannot be longer than the list of schemas in `items`. If this value is `true`, that is equivalent to the empty schema (anything goes).

classmethod list(*session*, *paginated=True*, *base_path=None*, *allow_unknown_params=False*, ***, *microversion=None*, ***params*)

This method is a generator which yields resource objects.

A re-implementation of *list()* that handles glances single, unpaginated list implementation.

Refer to *list()* for full documentation including parameter, exception and return type documentation.

openstack.image.v2.metadef_schema

The MetadefSchema Class

The `MetadefSchema` class inherits from *Resource*.

class openstack.image.v2.metadef_schema.**MetadefSchema**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

base_path = '/schemas/metadefs'

The base part of the URI for this resource.

allow_fetch = True

Allow get operation for this resource.

additional_properties

A boolean value that indicates allows users to add custom properties.

definitions

A set of definitions.

required

A list of required resources.

properties

Schema properties.

openstack.image.v2.task

The Task Class

The Task class inherits from *Resource*.

class openstack.image.v2.task.**Task**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'tasks'

Plural form of key for resource.

base_path = '/tasks'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_list = True

Allow list operation for this resource.

created_at

The date and time when the task was created.

expires_at

The date and time when the task is subject to removal.

input

A JSON object specifying the input parameters to the task.

message

Human-readable text, possibly an empty string, usually displayed in an error situation to provide more information about what has occurred.

owner_id

The ID of the owner, or project, of the task.

result

A JSON object specifying the outcome of the task.

schema

The URL for schema of the task.

status

The status of the task.

type

The type of task represented by this content.

updated_at

The date and time when the task was updated.

openstack.image.v2.service_info

The Store Class

The `Store` class inherits from *Resource*.

class openstack.image.v2.service_info.**Store**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'stores'

Plural form of key for resource.

base_path = '/info/stores'

The base part of the URI for this resource.

allow_list = True

Allow list operation for this resource.

description

Description of the store

is_default

default

properties

properties

delete_image(*session*, *image*, *, *ignore_missing=False*)

Delete image from store

Parameters

- **session** The session to use for making this request.
- **image** The value can be either the ID of an image or a *Image* instance.

Returns

The result of the `delete` if resource found, else None.

Raises

NotFoundException when ignore_missing if `False` and a nonexistent resource is attempted to be deleted.

The Import Info Class

The `Import` class inherits from *Resource*.

class openstack.image.v2.service_info.**Import**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

base_path = '/info/import'

The base part of the URI for this resource.

allow_fetch = True

Allow get operation for this resource.

import_methods

import methods

KeyManager Resources

openstack.key_manager.v1.container

The Container Class

The `Container` class inherits from *Resource*.

class openstack.key_manager.v1.container.**Container**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'containers'

Plural form of key for resource.

base_path = '/containers'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

container_ref

A URI for this container

container_id

The ID for this container

created_at

The timestamp when this container was created.

name

The name of this container

secret_refs

A list of references to secrets in this container

status

The status of this container

type

The type of this container

updated_at

The timestamp when this container was updated.

consumers

A party interested in this container.

openstack.key_manager.v1.order

The Order Class

The `Order` class inherits from *Resource*.

class openstack.key_manager.v1.order.**Order**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'orders'

Plural form of key for resource.

base_path = '/orders'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

created_at

Timestamp in ISO8601 format of when the order was created

creator_id

Keystone Id of the user who created the order

meta

A dictionary containing key-value parameters which specify the details of an order request

order_ref

A URI for this order

order_id

The ID of this order

secret_ref

Secret href associated with the order

secret_id

Secret ID associated with the order

sub_status

Metadata associated with the order

sub_status_message

Metadata associated with the order

updated_at

Timestamp in ISO8601 format of the last time the order was updated.

openstack.key_manager.v1.secret

The Secret Class

The `Secret` class inherits from *Resource*.

class openstack.key_manager.v1.secret.**Secret**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'secrets'

Plural form of key for resource.

base_path = '/secrets'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

algorithm

Metadata provided by a user or system for informational purposes

bit_length

Metadata provided by a user or system for informational purposes. Value must be greater than zero.

content_types

A list of content types

expires_at

Once this timestamp has past, the secret will no longer be available.

created_at

Timestamp of when the secret was created.

mode

The type/mode of the algorithm associated with the secret information.

name

The name of the secret set by the user

secret_ref

A URI to the sercret

secret_type

Used to indicate the type of secret being stored.

status

The status of this secret

updated_at

A timestamp when this secret was updated.

payload

The secrets data to be stored. payload_content_type must also be supplied if payload is included. (optional)

payload_content_type

The media type for the content of the payload. (required if payload is included)

payload_content_encoding

The encoding used for the payload to be able to include it in the JSON request. Currently only base64 is supported. (required if payload is encoded)

fetch(*session*, *requires_id=True*, *base_path=None*, *error_message=None*, *skip_cache=False*, ***kwargs*)

Get a remote resource based on this instance.

Parameters

- **session** (`Adapter`) The session to use for making this request.
- **requires_id** (*boolean*) A boolean indicating whether resource ID should be part of the requested URI.

- **base_path** (*str*) Base part of the URI for fetching resources, if different from *base_path*.
- **error_message** (*str*) An Error message to be returned if requested object does not exist.
- **skip_cache** (*bool*) A boolean indicating whether optional API cache should be skipped for this invocation.
- **resource_response_key** (*str*) Overrides the usage of self.resource_key when processing the response body.
- **microversion** (*str*) API version to override the negotiated one.
- **params** (*dict*) Additional parameters that can be consumed.

Returns
This `Resource` instance.

Raises
MethodNotSupported if `Resource.allow_fetch` is not set to `True`.

Raises
NotFoundException if the resource was not found.

Load Balancer Resources

openstack.load_balancer.v2.load_balancer

The LoadBalancer Class

The `LoadBalancer` class inherits from *Resource*.

`class openstack.load_balancer.v2.load_balancer.LoadBalancer(_synchronized=False, connection=None, **attrs)`

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (`openstack.connection.Connection`) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

`resource_key = 'loadbalancer'`

Singular form of key for resource.

`resources_key = 'loadbalancers'`

Plural form of key for resource.

`base_path = '/lbaas/loadbalancers'`

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

is_admin_state_up

The administrative state of the load balancer *Type: bool*

availability_zone

Name of the target Octavia availability zone

created_at

Timestamp when the load balancer was created

description

The load balancer description

flavor_id

The load balancer flavor ID

listeners

List of listeners associated with this load balancer

name

The load balancer name

operating_status

Operating status of the load balancer

pools

List of pools associated with this load balancer

project_id

The ID of the project this load balancer is associated with.

provider

Provider name for the load balancer.

provisioning_status

The provisioning status of this load balancer

updated_at

Timestamp when the load balancer was last updated

vip_address

VIP address of load balancer

vip_network_id
VIP netowrk ID

vip_port_id
VIP port ID

vip_subnet_id
VIP subnet ID

additional_vips
Additional VIPs

delete(*session*, *error_message=None*, ***kwargs*)
Delete the remote resource based on this instance.

Parameters

- **session** (Adapter) The session to use for making this request.
- **microversion** (*str*) API version to override the negotiated one.
- **kwargs** (*dict*) Parameters that will be passed to _prepare_request()

Returns
This Resource instance.

Raises
MethodNotSupported if Resource.allow_commit is not set to True.

Raises
NotFoundException if the resource was not found.

failover(*session*)
Failover load balancer.

Parameters
session The session to use for making this request.

Returns
None

The LoadBalancerStats Class

The LoadBalancerStats class inherits from *Resource*.

class openstack.load_balancer.v2.load_balancer.**LoadBalancerStats**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be

used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'stats'

Singular form of key for resource.

base_path = '/lbaas/loadbalancers/%(lb_id)s/stats'

The base part of the URI for this resource.

allow_create = False

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = False

Allow update operation for this resource.

allow_delete = False

Allow delete operation for this resource.

allow_list = False

Allow list operation for this resource.

lb_id

The ID of the load balancer.

active_connections

The currently active connections.

bytes_in

The total bytes received.

bytes_out

The total bytes sent.

request_errors

The total requests that were unable to be fulfilled.

total_connections

The total connections handled.

The LoadBalancerFailover Class

The `LoadBalancerFailover` class inherits from *Resource*.

class openstack.load_balancer.v2.load_balancer.**LoadBalancerFailover**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.

- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

base_path = '/lbaas/loadbalancers/%(lb_id)s/failover'

The base part of the URI for this resource.

allow_create = False

Allow create operation for this resource.

allow_fetch = False

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = False

Allow delete operation for this resource.

allow_list = False

Allow list operation for this resource.

allow_empty_commit = True

Commits happen without header or body being dirty.

requires_id = False

Do calls for this resource require an id

lb_id

The ID of the load balancer.

commit(*session*, *prepend_key=True*, *has_body=False*, **args*, ***kwargs*)

Commit the state of the instance to the remote resource.

Parameters

- **session** (Adapter) The session to use for making this request.
- **prepend_key** A boolean indicating whether the resource_key should be prepended in a resource update request. Default to True.
- **retry_on_conflict** (*bool*) Whether to enable retries on HTTP CONFLICT (409). Value of None leaves the *Adapter* defaults.
- **base_path** (*str*) Base part of the URI for modifying resources, if different from *base_path*.
- **microversion** (*str*) API version to override the negotiated one.
- **kwargs** (*dict*) Parameters that will be passed to _prepare_request()

Returns

This Resource instance.

Raises

MethodNotSupported if Resource.allow_commit is not set to True.

openstack.load_balancer.v2.listener

The Listener Class

The `Listener` class inherits from *Resource*.

class openstack.load_balancer.v2.listener.**Listener**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'listener'

Singular form of key for resource.

resources_key = 'listeners'

Plural form of key for resource.

base_path = '/lbaas/listeners'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

allowed_cidrs

List of IPv4 or IPv6 CIDRs.

alpn_protocols

List of ALPN protocols.

connection_limit

The maximum number of connections permitted for this load balancer. Default is infinite.

created_at

Timestamp when the listener was created.

default_pool

Default pool to which the requests will be routed.

default_pool_id

ID of default pool. Must have compatible protocol with listener.

default_tls_container_ref

A reference to a container of TLS secrets.

description

Description for the listener.

is_hsts_include_subdomains

Defines whether the *include_subdomains* directive is used for HSTS or not

hsts_max_age

Enables HTTP Strict Transport Security (HSTS) and sets the *max_age* directive to given value

is_hsts_preload

Defines whether the *hsts_preload* directive is used for HSTS or not

insert_headers

Dictionary of additional headers insertion into HTTP header.

is_admin_state_up

The administrative state of the listener, which is up `True` or down `False`. *Type: bool*

l7_policies

List of l7policies associated with this listener.

load_balancer_id

The ID of the parent load balancer.

load_balancers

List of load balancers associated with this listener. *Type: list of dicts which contain the load balancer IDs*

name

Name of the listener

operating_status

Operating status of the listener.

project_id

The ID of the project this listener is associated with.

protocol

The protocol of the listener, which is TCP, HTTP, HTTPS or TERMINATED_HTTPS.

protocol_port

Port the listener will listen to, e.g. 80.

provisioning_status

The provisioning status of this listener.

sni_container_refs

A list of references to TLS secrets. *Type: list*

updated_at

Timestamp when the listener was last updated.

timeout_client_data

Frontend client inactivity timeout in milliseconds.

timeout_member_connect

Backend member connection timeout in milliseconds.

timeout_member_data

Backend member inactivity timeout in milliseconds.

timeout_tcp_inspect

Time, in milliseconds, to wait for additional TCP packets for content inspection.

tls_ciphers

Stores a cipher string in OpenSSL format.

tls_versions

A lsit of TLS protocols to be used by the listener

The ListenerStats Class

The `ListenerStats` class inherits from *Resource*.

class openstack.load_balancer.v2.listener.**ListenerStats**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'stats'

Singular form of key for resource.

base_path = '/lbaas/listeners/%(listener_id)s/stats'

The base part of the URI for this resource.

allow_create = False

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = False

Allow update operation for this resource.

allow_delete = False

Allow delete operation for this resource.

allow_list = False

Allow list operation for this resource.

listener_id

The ID of the listener.

active_connections

The currently active connections.

bytes_in

The total bytes received.

bytes_out

The total bytes sent.

request_errors

The total requests that were unable to be fulfilled.

total_connections

The total connections handled.

openstack.load_balancer.v2.pool

The Pool Class

The Pool class inherits from *Resource*.

class openstack.load_balancer.v2.pool.**Pool**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'pool'

Singular form of key for resource.

resources_key = 'pools'

Plural form of key for resource.

base_path = '/lbaas/pools'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_commit = True

Allow update operation for this resource.

alpn_protocols

Properties List of ALPN protocols.

created_at

Timestamp when the pool was created

description

Description for the pool.

health_monitor_id

Health Monitor ID

is_admin_state_up

The administrative state of the pool *Type: bool*

lb_algorithm

The loadbalancing algorithm used in the pool

listener_id

ID of listener associated with this pool

listeners

List of listeners associated with this pool

loadbalancer_id

ID of load balancer associated with this pool

loadbalancers

List of loadbalancers associated with this pool

members

Members associated with this pool

name

The pool name

operating_status

Operating status of the pool

project_id

The ID of the project

protocol

The protocol of the pool

provisioning_status

Provisioning status of the pool

tls_ciphers

Stores a string of cipher strings in OpenSSL format.

session_persistence

A JSON object specifying the session persistence for the pool.

tls_versions

A list of TLS protocol versions to be used in by the pool

updated_at

Timestamp when the pool was updated

tls_enabled

Use TLS for connections to backend member servers *Type: bool*

ca_tls_container_ref

Stores the ca certificate used by backend servers

crl_container_ref

Stores the revocation list file

openstack.load_balancer.v2.member

The Member Class

The `Member` class inherits from *Resource*.

class openstack.load_balancer.v2.member.**Member**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'member'

Singular form of key for resource.

resources_key = 'members'

Plural form of key for resource.

base_path = '/lbaas/pools/%(pool_id)s/members'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

address

The IP address of the member.

created_at

Timestamp when the member was created.

is_admin_state_up

The administrative state of the member, which is up `True` or down `False`. *Type: bool*

monitor_address

IP address used to monitor this member

monitor_port

Port used to monitor this member

name

Name of the member.

operating_status

Operating status of the member.

pool_id

The ID of the owning pool.

provisioning_status

The provisioning status of this member.

project_id

The ID of the project this member is associated with.

protocol_port

The port on which the application is hosted.

subnet_id

Subnet ID in which to access this member.

updated_at

Timestamp when the member was last updated.

weight

A positive integer value that indicates the relative portion of traffic that this member should receive from the pool. For example, a member with a weight of 10 receives five times as much traffic as a member with weight of 2.

backup

A bool value that indicates whether the member is a backup or not. Backup members only receive traffic when all non-backup members are down.

openstack.load_balancer.v2.health_monitor

The HealthMonitor Class

The `HealthMonitor` class inherits from *Resource*.

class openstack.load_balancer.v2.health_monitor.**HealthMonitor**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'healthmonitor'

Singular form of key for resource.

resources_key = 'healthmonitors'

Plural form of key for resource.

base_path = '/lbaas/healthmonitors'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_commit = True

Allow update operation for this resource.

created_at

Properties Timestamp when the health monitor was created.

delay

The time, in seconds, between sending probes to members.

expected_codes

The expected http status codes to get from a successful health check

http_method

The HTTP method that the monitor uses for requests

is_admin_state_up

The administrative state of the health monitor *Type: bool*

max_retries

The number of successful checks before changing the operating status of the member to ONLINE.

max_retries_down

The number of allowed check failures before changing the operating status of the member to ERROR.

name

The health monitor name

operating_status

Operating status of the member.

pools

List of associated pools. *Type: list of dicts which contain the pool IDs*

pool_id

The ID of the associated Pool

project_id

The ID of the project

provisioning_status

The provisioning status of this member.

timeout

The time, in seconds, after which a health check times out

type

The type of health monitor

updated_at

Timestamp when the member was last updated.

url_path

The HTTP path of the request to test the health of a member

openstack.load_balancer.v2.l7_policy

The L7Policy Class

The `L7Policy` class inherits from *Resource*.

class openstack.load_balancer.v2.l7_policy.**L7Policy**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'l7policy'

Singular form of key for resource.

resources_key = 'l7policies'

Plural form of key for resource.

base_path = '/lbaas/l7policies'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

action

Properties The action to be taken l7policy is matched

created_at

Timestamp when the L7 policy was created.

description

The l7policy description

is_admin_state_up

The administrative state of the l7policy *Type: bool*

listener_id

The ID of the listener associated with this l7policy

name

The l7policy name

operating_status

Operating status of the member.

position

Sequence number of this l7policy

project_id

The ID of the project this l7policy is associated with.

provisioning_status

The provisioning status of this l7policy

redirect_pool_id

The ID of the pool to which the requests will be redirected

redirect_prefix

The URL prefix to which the requests should be redirected

redirect_url

The URL to which the requests should be redirected

rules

The list of L7Rules associated with the l7policy

updated_at

Timestamp when the member was last updated.

openstack.load_balancer.v2.l7_rule

The L7Rule Class

The `L7Rule` class inherits from *Resource*.

class openstack.load_balancer.v2.l7_rule.**L7Rule**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'rule'

Singular form of key for resource.

resources_key = 'rules'

Plural form of key for resource.

base_path = '/lbaas/l7policies/%(l7policy_id)s/rules'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

is_admin_state_up

Properties The administrative state of the l7policy *Type: bool*

compare_type

comparison type to be used with the value in this L7 rule.

created_at

Timestamp when the L7 rule was created.

key

The key to use for the comparison.

l7_policy_id

The ID of the associated l7 policy

operating_status

The operating status of this l7rule

project_id

The ID of the project this l7policy is associated with.

provisioning_status

The provisioning status of this l7policy

type

The type of L7 rule

updated_at

Timestamp when the L7 rule was updated.

rule_value

value to be compared with

openstack.load_balancer.v2.provider

The Provider Class

The `Provider` class inherits from *Resource*.

class openstack.load_balancer.v2.provider.**Provider**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.

- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'providers'

Plural form of key for resource.

base_path = '/lbaas/providers'

The base part of the URI for this resource.

allow_create = False

Allow create operation for this resource.

allow_fetch = False

Allow get operation for this resource.

allow_commit = False

Allow update operation for this resource.

allow_delete = False

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

name

The provider name.

description

The provider description.

The Provider Flavor Capabilities Class

The `ProviderFlavorCapabilities` class inherits from *Resource*.

class openstack.load_balancer.v2.provider.**ProviderFlavorCapabilities**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'flavor_capabilities'

Plural form of key for resource.

base_path = '/lbaas/providers/%(provider)s/flavor_capabilities'

The base part of the URI for this resource.

allow_create = False

Allow create operation for this resource.

allow_fetch = False

Allow get operation for this resource.

allow_commit = False

Allow update operation for this resource.

allow_delete = False

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

provider

The provider name to query.

name

The provider name.

description

The provider description.

openstack.load_balancer.v2.flavor_profile

The FlavorProfile Class

The `FlavorProfile` class inherits from *Resource*.

class openstack.load_balancer.v2.flavor_profile.**FlavorProfile**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'flavorprofile'

Singular form of key for resource.

resources_key = 'flavorprofiles'

Plural form of key for resource.

base_path = '/lbaas/flavorprofiles'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

id

The ID of the flavor profile.

name

The name of the flavor profile.

provider_name

The provider this flavor profile is for.

flavor_data

The JSON string containing the flavor metadata.

openstack.load_balancer.v2.flavor

The Flavor Class

The `Flavor` class inherits from *Resource*.

class openstack.load_balancer.v2.flavor.**Flavor**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'flavor'

Singular form of key for resource.

resources_key = 'flavors'

Plural form of key for resource.

base_path = '/lbaas/flavors'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

id

The ID of the flavor.

name

The name of the flavor.

description

The flavor description.

flavor_profile_id

The associated flavor profile ID

is_enabled

Whether the flavor is enabled for use or not.

openstack.load_balancer.v2.quota

The Quota Class

The Quota class inherits from *Resource*.

class openstack.load_balancer.v2.quota.**Quota**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of self._connection in Resource code should protect itself with a check for None.

resource_key = 'quota'

Singular form of key for resource.

resources_key = 'quotas'

Plural form of key for resource.

base_path = '/lbaas/quotas'

The base part of the URI for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

load_balancers

The maximum amount of load balancers you can have. *Type: int*

listeners

The maximum amount of listeners you can create. *Type: int*

pools

The maximum amount of pools you can create. *Type: int*

health_monitors

The maximum amount of health monitors you can create. *Type: int*

members

The maximum amount of members you can create. *Type: int*

project_id

The ID of the project this quota is associated with.

openstack.load_balancer.v2.amphora

The Amphora Class

The Amphora class inherits from *Resource*.

class openstack.load_balancer.v2.amphora.**Amphora**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of self._connection in Resource code should protect itself with a check for None.

resource_key = 'amphora'

Singular form of key for resource.

resources_key = 'amphorae'

Plural form of key for resource.

base_path = '/octavia/amphorae'

The base part of the URI for this resource.

allow_create = False

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = False

Allow update operation for this resource.

allow_delete = False

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

id

The ID of the amphora.

loadbalancer_id

The ID of the load balancer.

compute_id

The ID of the amphora resource in the compute system.

lb_network_ip

The management IP of the amphora.

vrrp_ip

The address of the vrrp port on the amphora.

ha_ip

The IP address of the Virtual IP (VIP).

vrrp_port_id

The vrrp ports ID in the networking system.

ha_port_id

The ID of the Virtual IP (VIP) port.

cert_expiration

The date the certificate for the amphora expires.

cert_busy

Whether the certificate is in the process of being replaced.

role

The role configured for the amphora. One of STANDALONE, MASTER, BACKUP.

status

The status of the amphora. One of: BOOTING, ALLOCATED, READY, PENDING_CREATE, PENDING_DELETE, DELETED, ERROR.

vrrp_interface

The bound interface name of the vrrp port on the amphora.

vrrp_id

The vrrp groups ID for the amphora.

vrrp_priority

The priority of the amphora in the vrrp group.

cached_zone

The availability zone of a compute instance, cached at create time.

created_at

The UTC date and timestamp when the resource was created.

updated_at

The UTC date and timestamp when the resource was last updated.

image_id

The ID of the glance image used for the amphora.

compute_flavor

The ID of the compute flavor used for the amphora.

configure(*session*)

Configure load balancer.

Update the amphora agent configuration. This will push the new configuration to the amphora agent and will update the configuration options that are mutatable.

Parameters
session The session to use for making this request.

Returns
None

failover(*session*)

Failover load balancer.

Parameters
session The session to use for making this request.

Returns
None

The AmphoraConfig Class

The `AmphoraConfig` class inherits from *Resource*.

class openstack.load_balancer.v2.amphora.**AmphoraConfig**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

base_path = '/octavia/amphorae/%(amphora_id)s/config'

The base part of the URI for this resource.

allow_create = False

Allow create operation for this resource.

allow_fetch = False

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = False

Allow delete operation for this resource.

allow_list = False

Allow list operation for this resource.

allow_empty_commit = True

Commits happen without header or body being dirty.

requires_id = False

Do calls for this resource require an id

amphora_id

The ID of the amphora.

commit(*session*, *prepend_key=True*, *has_body=False*, **args*, ***kwargs*)

Commit the state of the instance to the remote resource.

Parameters

- **session** (`Adapter`) The session to use for making this request.
- **prepend_key** A boolean indicating whether the resource_key should be prepended in a resource update request. Default to True.
- **retry_on_conflict** (*bool*) Whether to enable retries on HTTP CONFLICT (409). Value of `None` leaves the *Adapter* defaults.
- **base_path** (*str*) Base part of the URI for modifying resources, if different from *base_path*.
- **microversion** (*str*) API version to override the negotiated one.
- **kwargs** (*dict*) Parameters that will be passed to _prepare_request()

Returns

This `Resource` instance.

Raises

MethodNotSupported if `Resource.allow_commit` is not set to `True`.

The AmphoraFailover Class

The `AmphoraFailover` class inherits from *Resource*.

class openstack.load_balancer.v2.amphora.**AmphoraFailover**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (*openstack.connection.Connection*) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

base_path = '/octavia/amphorae/%(amphora_id)s/failover'

The base part of the URI for this resource.

allow_create = False

Allow create operation for this resource.

allow_fetch = False

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = False

Allow delete operation for this resource.

allow_list = False

Allow list operation for this resource.

allow_empty_commit = True

Commits happen without header or body being dirty.

requires_id = False

Do calls for this resource require an id

amphora_id

The ID of the amphora.

commit(*session*, *prepend_key=True*, *has_body=False*, **args*, ***kwargs*)

Commit the state of the instance to the remote resource.

Parameters

- **session** (`Adapter`) The session to use for making this request.
- **prepend_key** A boolean indicating whether the resource_key should be prepended in a resource update request. Default to True.

- **retry_on_conflict** (*bool*) Whether to enable retries on HTTP CONFLICT (409). Value of `None` leaves the *Adapter* defaults.
- **base_path** (*str*) Base part of the URI for modifying resources, if different from *base_path*.
- **microversion** (*str*) API version to override the negotiated one.
- **kwargs** (*dict*) Parameters that will be passed to _prepare_request()

Returns
This `Resource` instance.

Raises
MethodNotSupported if `Resource.allow_commit` is not set to `True`.

openstack.load_balancer.v2.availability_zone_profile

The AvailabilityZoneProfile Class

The `AvailabilityZoneProfile` class inherits from *Resource*.

class openstack.load_balancer.v2.availability_zone_profile.**AvailabilityZoneProfile**(*_synchroni con- nec- tion=None **at- trs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'availability_zone_profile'
Singular form of key for resource.

resources_key = 'availability_zone_profiles'
Plural form of key for resource.

base_path = '/lbaas/availabilityzoneprofiles'
The base part of the URI for this resource.

allow_create = True
Allow create operation for this resource.

allow_fetch = True
Allow get operation for this resource.

allow_commit = True
Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

id

The ID of the availability zone profile.

name

The name of the availability zone profile.

provider_name

The provider this availability zone profile is for.

availability_zone_data

The JSON string containing the availability zone metadata.

openstack.load_balancer.v2.availability_zone

The AvailabilityZone Class

The `AvailabilityZone` class inherits from *Resource*.

class openstack.load_balancer.v2.availability_zone.**AvailabilityZone**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'availability_zone'

Singular form of key for resource.

resources_key = 'availability_zones'

Plural form of key for resource.

base_path = '/lbaas/availabilityzones'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

name

The name of the availability zone.

description

The availability zone description.

availability_zone_profile_id

The associated availability zone profile ID

is_enabled

Whether the availability zone is enabled for use or not.

Network Resources

openstack.network.v2.address_group

The AddressGroup Class

The `AddressGroup` class inherits from *Resource*.

class openstack.network.v2.address_group.**AddressGroup**(*_synchronized=False*, *connection=None*, ***attrs*)

Address group extension.

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'address_group'

Singular form of key for resource.

resources_key = 'address_groups'

Plural form of key for resource.

base_path = '/address-groups'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

id

The ID of the address group.

name

The address group name.

description

The address group name.

project_id

The ID of the project that owns the address group.

tenant_id

Tenant_id (deprecated attribute).

addresses

The IP addresses of the address group.

add_addresses(*session*, *addresses*)

Add addresses into the address group.

Parameters

- **session** (Adapter) The session to communicate through.
- **addresses** (*list*) The list of address strings.

Returns

The response as a AddressGroup object with updated addresses

Raises

SDKException on error.

remove_addresses(*session*, *addresses*)

Remove addresses from the address group.

Parameters

- **session** (Adapter) The session to communicate through.
- **addresses** (*list*) The list of address strings.

Returns

The response as a AddressGroup object with updated addresses

Raises

SDKException on error.

openstack.network.v2.address_scope

The AddressScope Class

The `AddressScope` class inherits from *Resource*.

class openstack.network.v2.address_scope.**AddressScope**(*_synchronized=False*, *connection=None*, ***attrs*)

Address scope extension.

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'address_scope'

Singular form of key for resource.

resources_key = 'address_scopes'

Plural form of key for resource.

base_path = '/address-scopes'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

name

The address scope name.

project_id

The ID of the project that owns the address scope.

tenant_id

Tenant_id (deprecated attribute).

ip_version

The IP address family of the address scope. *Type: int*

is_shared

Indicates whether this address scope is shared across all projects. *Type: bool*

openstack.network.v2.agent

The Agent Class

The `Agent` class inherits from *Resource*.

class `openstack.network.v2.agent.`**Agent**(*_synchronized=False*, *connection=None*, ***attrs*)

Neutron agent extension.

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (*openstack.connection.Connection*) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'agent'

Singular form of key for resource.

resources_key = 'agents'

Plural form of key for resource.

base_path = '/agents'

The base part of the URI for this resource.

allow_create = False

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

agent_type

The type of network agent.

availability_zone

Availability zone for the network agent.

binary

The name of the network agents application binary.

configuration

Network agent configuration data specific to the agent_type.

created_at

Timestamp when the network agent was created.

description

The network agent description.

last_heartbeat_at

Timestamp when the network agents heartbeat was last seen.

host

The host the agent is running on.

is_admin_state_up

The administrative state of the network agent, which is up `True` or down `False`. *Type: bool*

is_alive

Whether or not the network agent is alive. *Type: bool*

resources_synced

Whether or not the agent is succesffully synced towards placement. Agents supporting the guaranteed minimum bandwidth feature share their resource view with neutron-server and neutron-server share this view with placement, resources_synced represents the success of the latter. The value None means no resource view synchronization to Placement was attempted. true / false values signify the success of the last synchronization attempt. *Type: bool*

started_at

Timestamp when the network agent was last started.

topic

The messaging queue topic the network agent subscribes to.

ha_state

The HA state of the L3 agent. This is one of active, standby or fault for HA routers, or None for other types of routers.

get_bgp_speakers_hosted_by_dragent(*session*)

List BGP speakers hosted by a Dynamic Routing Agent

Parameters
session (`Adapter`) The session to communicate through.

Returns
A list of BgpSpeakers

Return type
BgpSpeaker

openstack.network.v2.auto_allocated_topology

The Auto Allocated Topology Class

The `Auto Allocated Toplogy` class inherits from *Resource*.

class openstack.network.v2.auto_allocated_topology.**AutoAllocatedTopology**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'auto_allocated_topology'

Singular form of key for resource.

base_path = '/auto-allocated-topology'

The base part of the URI for this resource.

allow_create = False

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = False

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = False

Allow list operation for this resource.

project_id

Project ID If project is not specified the topology will be created for project user is authenticated against. Will return in error if resources have not been configured correctly To use this feature auto-allocated-topology, subnet_allocation, external-net and router extensions must be enabled and set up.

tenant_id

Tenant_id (deprecated attribute).

openstack.network.v2.availability_zone

The AvailabilityZone Class

The `AvailabilityZone` class inherits from *Resource*.

class openstack.network.v2.availability_zone.**AvailabilityZone**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'availability_zone'

Singular form of key for resource.

resources_key = 'availability_zones'

Plural form of key for resource.

base_path = '/availability_zones'

The base part of the URI for this resource.

allow_create = False

Allow create operation for this resource.

allow_fetch = False

Allow get operation for this resource.

allow_commit = False

Allow update operation for this resource.

allow_delete = False

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

name

Name of the availability zone.

resource

Type of resource for the availability zone, such as `network`.

state

State of the availability zone, either `available` or `unavailable`.

openstack.network.v2.bgp_peer

The BgpPeer Class

The `BgpPeer` class inherits from *Resource*.

class openstack.network.v2.bgp_peer.**BgpPeer**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'bgp_peer'

Singular form of key for resource.

resources_key = 'bgp_peers'

Plural form of key for resource.

base_path = '/bgp-peers'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

id

The Id of the BGP Peer

name

The BGP Peers name.

project_id

The ID of the project that owns the BGP Peer

tenant_id

Tenant_id (deprecated attribute).

auth_type

The authentication type for the BGP Peer, can be none or md5. none by default.

remote_as

The remote Autonomous System number of the BGP Peer.

peer_ip

The ip address of the Peer.

openstack.network.v2.bgp_speaker

The BgpSpeaker Class

The BgpSpeaker class inherits from *Resource*.

class openstack.network.v2.bgp_speaker.**BgpSpeaker**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'bgp_speaker'

Singular form of key for resource.

resources_key = 'bgp_speakers'

Plural form of key for resource.

base_path = '/bgp-speakers'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

id

The Id of the BGP Speaker

name

The BGP speakers name.

project_id

The ID of the project that owns the BGP Speaker.

tenant_id

Tenant_id (deprecated attribute).

ip_version

The IP version (4 or 6) of the BGP Speaker.

advertise_floating_ip_host_routes

Whether to enable or disable the advertisement of floating ip host routes by the BGP Speaker. True by default.

advertise_tenant_networks

Whether to enable or disable the advertisement of tenant network routes by the BGP Speaker. True by default.

local_as

The local Autonomous System number of the BGP Speaker.

networks

The ID of the network to which the BGP Speaker is associated.

add_bgp_peer(*session*, *peer_id*)

Add BGP Peer to a BGP Speaker

Parameters

- **session** (Adapter) The session to communicate through.
- **peer_id** id of the peer to associate with the speaker.

Returns

A dictionary as the API Reference describes it.

Raises

SDKException on error.

remove_bgp_peer(*session*, *peer_id*)

Remove BGP Peer from a BGP Speaker

Parameters

- **session** (Adapter) The session to communicate through.
- **peer_id** The ID of the peer to disassociate from the speaker.

Raises

SDKException on error.

add_gateway_network(*session*, *network_id*)

Add Network to a BGP Speaker

Param

session: The session to communicate through.

Parameters

network_id The ID of the network to associate with the speaker

Returns

A dictionary as the API Reference describes it.

remove_gateway_network(*session*, *network_id*)

Delete Network from a BGP Speaker

Parameters

- **session** (Adapter) The session to communicate through.
- **network_id** The ID of the network to disassociate from the speaker

get_advertised_routes(*session*)

List routes advertised by a BGP Speaker

Parameters

session (Adapter) The session to communicate through.

Returns

The response as a list of routes (cidr/nexthop pair advertised by the BGP Speaker.

Raises

SDKException on error.

get_bgp_dragents(*session*)

List Dynamic Routing Agents hosting a specific BGP Speaker

Parameters

session (Adapter) The session to communicate through.

Returns

The response as a list of dragents hosting a specific BGP Speaker.

Return type

Agent

Raises

SDKException on error.

add_bgp_speaker_to_dragent(*session*, *bgp_agent_id*)

Add BGP Speaker to a Dynamic Routing Agent

Parameters

- **session** (Adapter) The session to communicate through.
- **bgp_agent_id** The id of the dynamic routing agent to which add the speaker.

remove_bgp_speaker_from_dragent(*session*, *bgp_agent_id*)

Delete BGP Speaker from a Dynamic Routing Agent

Parameters

- **session** (Adapter) The session to communicate through.
- **bgp_agent_id** The id of the dynamic routing agent from which remove the speaker.

openstack.network.v2.bgpvpn

The BgpVpn Class

The BgpVpn class inherits from *Resource*.

class openstack.network.v2.bgpvpn.**BgpVpn**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'bgpvpn'

Singular form of key for resource.

resources_key = 'bgpvpns'

Plural form of key for resource.

base_path = '/bgpvpn/bgpvpns'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

id

The Id of the BGPVPN

name

The BGPVPNs name.

project_id

The ID of the project that owns the BGPVPN

tenant_id

Tenant_id (deprecated attribute).

route_distinguishers

List of route distinguisher strings.

route_targets

Route Targets that will be both imported and used for export.

import_targets
Additional Route Targets that will be imported.

export_targets
Additional Route Targets that will be used for export.

local_pref
The default BGP LOCAL_PREF of routes that will be advertised to the BGPVPN.

vni
The globally-assigned VXLAN vni for the BGP VPN.

type
Selection of the type of VPN and the technology behind it. Allowed values are l2 or l3.

openstack.network.v2.bgpvpn_network_association

The BgpVpnNetworkAssociation Class

The `BgpVpnNetworkAssociation` class inherits from *Resource*.

class openstack.network.v2.bgpvpn_network_association.**BgpVpnNetworkAssociation**(*_synchronized=True*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'network_association'
Singular form of key for resource.

resources_key = 'network_associations'
Plural form of key for resource.

base_path = '/bgpvpn/bgpvpns/%(bgpvpn_id)s/network_associations'
The base part of the URI for this resource.

allow_create = True
Allow create operation for this resource.

allow_fetch = True
Allow get operation for this resource.

allow_commit = False
Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

id

The Id of the BGPVPN

bgpvpn_id

The ID of the BGPVPN who owns Network Association.

project_id

The ID of the project that owns the BGPVPN

tenant_id

Tenant_id (deprecated attribute).

network_id

The ID of a Neutron network with which to associate the BGP VPN.

openstack.network.v2.bgpvpn_port_association

The BgpVpnPortAssociation Class

The BgpVpnPortAssociation class inherits from *Resource*.

class openstack.network.v2.bgpvpn_port_association.**BgpVpnPortAssociation**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'port_association'

Singular form of key for resource.

resources_key = 'port_associations'

Plural form of key for resource.

base_path = '/bgpvpn/bgpvpns/%(bgpvpn_id)s/port_associations'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

id

The Id of the BGPVPN

bgpvpn_id

The ID of the BGPVPN who owns Network Association.

project_id

The ID of the project that owns the BGPVPN

tenant_id

Tenant_id (deprecated attribute).

port_id

The ID of a Neutron Port with which to associate the BGP VPN.

advertise_fixed_ips

Boolean flag controlling whether or not the fixed IPs of a port will be advertised to the BGPVPN (default: true).

routes

List of routes, each route being a dict with at least a type key, which can be prefix or bgpvpn. For the prefix type, the IP prefix (v4 or v6) to advertise is specified in the prefix key. For the bgpvpn type, the bgpvpn_id key specifies the BGPVPN from which routes will be readvertised

openstack.network.v2.bgpvpn_router_association

The BgpVpnRouterAssociation Class

The `BgpVpnRouterAssociation` class inherits from *Resource*.

class openstack.network.v2.bgpvpn_router_association.**BgpVpnRouterAssociation**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.

- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'router_association'

Singular form of key for resource.

resources_key = 'router_associations'

Plural form of key for resource.

base_path = '/bgpvpn/bgpvpns/%(bgpvpn_id)s/router_associations'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

id

The Id of the BGPVPN

bgpvpn_id

The ID of the BGPVPN who owns Network Association.

project_id

The ID of the project that owns the BGPVPN

tenant_id

Tenant_id (deprecated attribute).

router_id

The ID of a Neutron router with which to associate the BGP VPN.

advertise_extra_routes

Boolean flag controlling whether or not the routes specified in the routes attribute of the router will be advertised to the BGPVPN (default: true).

openstack.network.v2.extension

The Extension Class

The `Extension` class inherits from *Resource*.

class openstack.network.v2.extension.**Extension**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'extension'

Singular form of key for resource.

resources_key = 'extensions'

Plural form of key for resource.

base_path = '/extensions'

The base part of the URI for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_list = True

Allow list operation for this resource.

alias

An alias the extension is known under.

description

Text describing what the extension does.

links

Links pertaining to this extension.

name

The name of this extension.

updated_at

Timestamp when the extension was last updated.

openstack.network.v2.flavor

The Flavor Class

The `Flavor` class inherits from *Resource*.

class openstack.network.v2.flavor.**Flavor**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'flavor'

Singular form of key for resource.

resources_key = 'flavors'

Plural form of key for resource.

base_path = '/flavors'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

description

description for the flavor

is_enabled

Sets enabled flag

name

The name of the flavor

service_type

Service type to which the flavor applies

service_profile_ids

IDs of service profiles associated with this flavor

openstack.network.v2.floating_ip

The FloatingIP Class

The `FloatingIP` class inherits from *Resource*.

class openstack.network.v2.floating_ip.**FloatingIP**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'floatingip'

Singular form of key for resource.

resources_key = 'floatingips'

Plural form of key for resource.

base_path = '/floatingips'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

created_at

Timestamp at which the floating IP was created.

description

The floating IP description.

dns_domain

The DNS domain.

dns_name

The DNS name.

fixed_ip_address

The fixed IP address associated with the floating IP. If you intend to associate the floating IP with a fixed IP at creation time, then you must indicate the identifier of the internal port. If an internal port has multiple associated IP addresses, the service chooses the first IP unless you explicitly specify the parameter fixed_ip_address to select a specific IP.

floating_ip_address

The floating IP address.

name

Floating IP object doesnt have name attribute, set ip address to name so that user could find floating IP by UUID or IP address using find_ip

floating_network_id

The ID of the network associated with the floating IP.

port_details

Read-only. The details of the port that this floating IP associates with. Present if `fip-port-details` extension is loaded. *Type: dict with keys: name, network_id, mac_address, admin_state_up, status, device_id, device_owner*

port_id

The port ID.

qos_policy_id

The ID of the QoS policy attached to the floating IP.

project_id

The ID of the project this floating IP is associated with.

tenant_id

Tenant_id (deprecated attribute).

router_id

The ID of an associated router.

status

The floating IP status. Value is `ACTIVE` or `DOWN`.

updated_at

Timestamp at which the floating IP was last updated.

subnet_id

The Subnet ID associated with the floating IP.

openstack.network.v2.health_monitor

The HealthMonitor Class

The `HealthMonitor` class inherits from *Resource*.

class openstack.network.v2.health_monitor.**HealthMonitor**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'healthmonitor'

Singular form of key for resource.

resources_key = 'healthmonitors'

Plural form of key for resource.

base_path = '/lbaas/healthmonitors'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

delay

The time, in seconds, between sending probes to members.

expected_codes

Expected HTTP codes for a passing HTTP(S) monitor.

http_method

The HTTP method that the monitor uses for requests.

is_admin_state_up

The administrative state of the health monitor, which is up `True` or down `False`. *Type: bool*

max_retries

Maximum consecutive health probe tries.

name

Name of the health monitor.

pool_ids

List of pools associated with this health monitor *Type: list of dicts which contain the pool IDs*

pool_id

The ID of the pool associated with this health monitor

project_id

The ID of the project this health monitor is associated with.

tenant_id

Tenant_id (deprecated attribute).

timeout

The maximum number of seconds for a monitor to wait for a connection to be established before it times out. This value must be less than the delay value.

type

The type of probe sent by the load balancer to verify the member state, which is PING, TCP, HTTP, or HTTPS.

url_path

Path portion of URI that will be probed if type is HTTP(S).

openstack.network.v2.listener

The Listener Class

The `Listener` class inherits from *Resource*.

class openstack.network.v2.listener.**Listener**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'listener'

Singular form of key for resource.

resources_key = 'listeners'

Plural form of key for resource.

base_path = '/lbaas/listeners'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

connection_limit

The maximum number of connections permitted for this load balancer. Default is infinite.

default_pool_id

ID of default pool. Must have compatible protocol with listener.

default_tls_container_ref

A reference to a container of TLS secrets.

description

Description for the listener.

is_admin_state_up

The administrative state of the listener, which is up `True` or down `False`. *Type: bool*

load_balancer_ids

List of load balancers associated with this listener. *Type: list of dicts which contain the load balancer IDs*

load_balancer_id

The ID of the load balancer associated with this listener.

name

Name of the listener

project_id

The ID of the project this listener is associated with.

protocol

The protocol of the listener, which is TCP, HTTP, HTTPS or TERMINATED_HTTPS.

protocol_port

Port the listener will listen to, e.g. 80.

sni_container_refs

A list of references to TLS secrets. *Type: list*

openstack.network.v2.load_balancer

The LoadBalancer Class

The `LoadBalancer` class inherits from *Resource*.

class openstack.network.v2.load_balancer.**LoadBalancer**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'loadbalancer'

Singular form of key for resource.

resources_key = 'loadbalancers'

Plural form of key for resource.

base_path = '/lbaas/loadbalancers'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

description

Description for the load balancer.

is_admin_state_up

The administrative state of the load balancer, which is up `True` or down `False`. *Type: bool*

listener_ids

List of listeners associated with this load balancer. *Type: list of dicts which contain the listener IDs*

name

Name of the load balancer

operating_status

Status of load_balancer operating, e.g. ONLINE, OFFLINE.

pool_ids

List of pools associated with this load balancer. *Type: list of dicts which contain the pool IDs*

project_id

The ID of the project this load balancer is associated with.

tenant_id

Tenant_id (deprecated attribute).

provider

The name of the provider.

provisioning_status

Status of load balancer provisioning, e.g. ACTIVE, INACTIVE.

vip_address

The IP address of the VIP.

vip_port_id

The ID of the port for the VIP.

vip_subnet_id

The ID of the subnet on which to allocate the VIP address.

openstack.network.v2.local_ip

The LocalIP Class

The `LocalIP` class inherits from *Resource*.

class openstack.network.v2.local_ip.**LocalIP**(*_synchronized=False*, *connection=None*, ***attrs*)

Local IP extension.

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'local_ip'

Singular form of key for resource.

resources_key = 'local_ips'

Plural form of key for resource.

base_path = '/local_ips'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

created_at
Timestamp at which the floating IP was created.

description
The local ip description.

id
The ID of the local ip.

ip_mode
The local ip ip-mode.

local_ip_address
The Local IP address.

local_port_id
The ID of the port that owns the local ip.

name
The local ip name.

network_id
The ID of the network that owns the local ip.

project_id
The ID of the project that owns the local ip.

revision_number
The local ip revision number.

updated_at
Timestamp at which the floating IP was last updated.

openstack.network.v2.local_ip_association

The LocalIPAssociation Class

The `LocalIPAssociation` class inherits from *Resource*.

class openstack.network.v2.local_ip_association.**LocalIPAssociation**(*_synchronized=False*, *connection=None*, ***attrs*)

Local IP extension.

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'port_association'

Singular form of key for resource.

resources_key = 'port_associations'

Plural form of key for resource.

base_path = '/local_ips/%(local_ip_id)s/port_associations'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

fixed_port_id

The fixed port ID.

fixed_ip

The fixed IP.

host

Host

local_ip_address

The local ip address

local_ip_id

The ID of Local IP address

openstack.network.v2.metering_label

The MeteringLabel Class

The `MeteringLabel` class inherits from *Resource*.

class openstack.network.v2.metering_label.**MeteringLabel**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be

used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'metering_label'

Singular form of key for resource.

resources_key = 'metering_labels'

Plural form of key for resource.

base_path = '/metering/metering-labels'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

description

Description of the metering label.

name

Name of the metering label.

project_id

The ID of the project this metering label is associated with.

tenant_id

Tenant_id (deprecated attribute).

is_shared

Indicates whether this label is shared across all tenants. *Type: bool*

openstack.network.v2.metering_label_rule

The MeteringLabelRule Class

The `MeteringLabelRule` class inherits from *Resource*.

class openstack.network.v2.metering_label_rule.**MeteringLabelRule**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'metering_label_rule'

Singular form of key for resource.

resources_key = 'metering_label_rules'

Plural form of key for resource.

base_path = '/metering/metering-label-rules'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

direction

ingress or egress: The direction in which metering label rule is applied. Default: `"ingress"`

is_excluded

Specify whether the `remote_ip_prefix` will be excluded or not from traffic counters of the metering label, ie: to not count the traffic of a specific IP address of a range. Default: `False`, *Type: bool*

metering_label_id

The metering label ID to associate with this metering label rule.

project_id

The ID of the project this metering label rule is associated with.

tenant_id

Tenant_id (deprecated attribute).

remote_ip_prefix

The remote IP prefix to be associated with this metering label rule.

source_ip_prefix

The source IP prefix to be associated with this metering label rule.

destination_ip_prefix

The destination IP prefix to be associated with this metering label rule

openstack.network.v2.ndp_proxy

The NDPProxy Class

The NDPProxy class inherits from *Resource*.

class openstack.network.v2.ndp_proxy.**NDPProxy**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'ndp_proxy'

Singular form of key for resource.

resources_key = 'ndp_proxies'

Plural form of key for resource.

base_path = '/ndp_proxies'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

created_at

Timestamp at which the NDP proxy was created.

description

The description

id

The ID of the NDP proxy.

ip_address

The internal IP address

name

The name of this resource.

port_id

The ID of internal port

project_id

The ID of the project that owns the NDP proxy.

revision_number

The NDP proxy revision number.

router_id

The ID of Router

updated_at

Timestamp at which the NDP proxy was last updated.

openstack.network.v2.network

The Network Class

The `Network` class inherits from *Resource*.

class openstack.network.v2.network.**Network**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'network'

Singular form of key for resource.

resources_key = 'networks'

Plural form of key for resource.

base_path = '/networks'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

availability_zone_hints

Availability zone hints to use when scheduling the network. *Type: list of availability zone names*

availability_zones

Availability zones for the network. *Type: list of availability zone names*

created_at

Timestamp when the network was created.

description

The network description.

dns_domain

The DNS domain associated.

ipv4_address_scope_id

The ID of the IPv4 address scope for the network.

ipv6_address_scope_id

The ID of the IPv6 address scope for the network.

is_admin_state_up

The administrative state of the network, which is up `True` or down `False`. *Type: bool*

is_default

Whether or not this is the default external network. *Type: bool*

is_port_security_enabled

The port security status, which is enabled `True` or disabled `False`. *Type: bool Default: False* Available for multiple provider extensions.

is_router_external

Whether or not the router is external. *Type: bool Default: False*

is_shared

Indicates whether this network is shared across all tenants. By default, only administrative users can change this value. *Type: bool*

mtu

Read-only. The maximum transmission unit (MTU) of the network resource.

name

The network name.

project_id

The ID of the project this network is associated with.

provider_network_type

The type of physical network that maps to this network resource. For example, `flat`, `vlan`, `vxlan`, or `gre`. Available for multiple provider extensions.

provider_physical_network

The physical network where this network object is implemented. Available for multiple provider extensions.

provider_segmentation_id

An isolated segment ID on the physical network. The provider network type defines the segmentation model. Available for multiple provider extensions.

qos_policy_id

The ID of the QoS policy attached to the port.

segments

A list of provider segment objects. Available for multiple provider extensions.

status

The network status.

subnet_ids

The associated subnet IDs. *Type: list of strs of the subnet IDs*

updated_at

Timestamp when the network was last updated.

is_vlan_transparent

Indicates the VLAN transparency mode of the network

is_vlan_qinq

Indicates the VLAN QinQ mode of the network

openstack.network.v2.network_ip_availability

The NetworkIPAvailability Class

The `NetworkIPAvailability` class inherits from *Resource*.

class openstack.network.v2.network_ip_availability.**NetworkIPAvailability**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'network_ip_availability'

Singular form of key for resource.

resources_key = 'network_ip_availabilities'

Plural form of key for resource.

base_path = '/network-ip-availabilities'

The base part of the URI for this resource.

allow_create = False

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = False

Allow update operation for this resource.

allow_delete = False

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

network_id

Network ID to use when listing network IP availability.

network_name

Network Name for the particular network IP availability.

subnet_ip_availability

The Subnet IP Availability of all subnets of a network. *Type: list*

project_id

The ID of the project this network IP availability is associated with.

tenant_id

Tenant_id (deprecated attribute).

total_ips

The total ips of a network. *Type: int*

used_ips

The used or consumed ip of a network *Type: int*

openstack.network.v2.network_segment_range

The NetworkSegmentRange Class

The `NetworkSegmentRange` class inherits from `Resource`.

class openstack.network.v2.network_segment_range.**NetworkSegmentRange**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'network_segment_range'

Singular form of key for resource.

resources_key = 'network_segment_ranges'

Plural form of key for resource.

base_path = '/network_segment_ranges'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

name

The network segment range name.

default

The network segment range is loaded from the host configuration file. *Type: bool*

shared

The network segment range is shared with other projects. *Type: bool*

project_id

The ID of the project associated with this network segment range.

network_type

The type of network associated with this network segment range, such as `geneve`, `gre`, `vlan` or `vxlan`.

physical_network

The name of the physical network associated with this network segment range.

minimum

The minimum segmentation ID for this network segment range. The network type defines the segmentation model, VLAN ID for `vlan` network type and tunnel ID for `geneve`, `gre` and `vxlan` network types. *Type: int*

maximum

The maximum segmentation ID for this network segment range. The network type defines the segmentation model, VLAN ID for `vlan` network type and tunnel ID for `geneve`, `gre` and `vxlan` network types. *Type: int*

used

Mapping of which segmentation ID in the range is used by which tenant. *Type: dict*

available

List of available segmentation IDs in this network segment range. *Type: list*

openstack.network.v2.pool

The Pool Class

The `Pool` class inherits from *Resource*.

class openstack.network.v2.pool.**Pool**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'pool'

Singular form of key for resource.

resources_key = 'pools'

Plural form of key for resource.

base_path = '/lbaas/pools'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

description

Description for the pool.

health_monitor_id

The ID of the associated health monitors.

health_monitor_ids

The ID of the associated health monitors (LBaaS v1).

health_monitor_status

The statuses of the associated health monitors.

is_admin_state_up

The administrative state of the pool, which is up `True` or down `False`. *Type: bool*

lb_algorithm

The load-balancer algorithm, which is round-robin, least-connections, and so on. This value, which must be supported, is dependent on the load-balancer provider. Round-robin must be supported.

listener_ids

List of associated listeners. *Type: list of dicts which contain the listener IDs*

listener_id

ID of listener associated with this pool

load_balancer_ids

List of associated load balancers. *Type: list of dicts which contain the load balancer IDs*

load_balancer_id

ID of load balancer associated with this pool

member_ids

List of members that belong to the pool. *Type: list of dicts which contain the member IDs*

name

Pool name. Does not have to be unique.

project_id

The ID of the project this pool is associated with.

tenant_id

Tenant_id (deprecated attribute).

protocol

The protocol of the pool, which is TCP, HTTP, or HTTPS.

provider

The provider name of the load balancer service.

status

Human readable description of the status.

status_description

The status of the network.

subnet_id

The subnet on which the members of the pool will be located.

session_persistence

Session persistence algorithm that should be used (if any). *Type: dict with keys "type" and "cookie_name"*

virtual_ip_id

The ID of the virtual IP (VIP) address.

openstack.network.v2.pool_member

The PoolMember Class

The `PoolMember` class inherits from *Resource*.

class openstack.network.v2.pool_member.**PoolMember**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'member'

Singular form of key for resource.

resources_key = 'members'

Plural form of key for resource.

base_path = '/lbaas/pools/%(pool_id)s/members'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

pool_id

The ID of the owning pool

address

The IP address of the pool member.

is_admin_state_up

The administrative state of the pool member, which is up `True` or down `False`. *Type: bool*

name

Name of the pool member.

project_id

The ID of the project this pool member is associated with.

tenant_id

Tenant_id (deprecated attribute).

protocol_port

The port on which the application is hosted.

subnet_id

Subnet ID in which to access this pool member.

weight

A positive integer value that indicates the relative portion of traffic that this member should receive from the pool. For example, a member with a weight of 10 receives five times as much traffic as a member with weight of 2.

openstack.network.v2.port

The Port Class

The `Port` class inherits from *Resource*.

class openstack.network.v2.port.**Port**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'port'

Singular form of key for resource.

resources_key = 'ports'

Plural form of key for resource.

base_path = '/ports'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

allowed_address_pairs

Allowed address pairs list. Dictionary key `ip_address` is required and key `mac_address` is optional.

binding_host_id

The ID of the host where the port is allocated. In some cases, different implementations can run on different hosts.

binding_profile

A dictionary the enables the application running on the specified host to pass and receive vif port-specific information to the plug-in. *Type: dict*

binding_vif_details

Read-only. A dictionary that enables the application to pass information about functions that the Networking API provides. To enable or disable port filtering features such as security group and anti-MAC/IP spoofing, specify `port_filter: True` or `port_filter: False`. *Type: dict*

binding_vif_type

Read-only. The vif type for the specified port.

binding_vnic_type

The vnic type that is bound to the neutron port.

In POST and PUT operations, specify a value of `normal` (virtual nic), `direct` (pci passthrough), or `macvtap` (virtual interface with a tap-like software interface). These values support SR-IOV PCI passthrough networking. The ML2 plug-in supports the vnic_type.

In GET operations, the binding:vnic_type extended attribute is visible to only port owners and administrative users.

created_at

Timestamp when the port was created.

data_plane_status

Underlying data plane status of this port.

description

The port description.

device_id

Device ID of this port.

device_owner

Device owner of this port (e.g. `network:dhcp`).

dns_assignment
DNS assignment for the port.

dns_domain
DNS domain assigned to the port.

dns_name
DNS name for the port.

extra_dhcp_opts
Extra DHCP options.

fixed_ips
IP addresses for the port. Includes the IP address and subnet ID.

is_admin_state_up
The administrative state of the port, which is up `True` or down `False`. *Type: bool*

is_port_security_enabled
The port security status, which is enabled `True` or disabled `False`. *Type: bool Default: False*

mac_address
The MAC address of an allowed address pair.

name
The port name.

network_id
The ID of the attached network.

numa_affinity_policy
The NUMA affinity policy defined for this port.

project_id
The ID of the project who owns the network. Only administrative users can specify a project ID other than their own.

tenant_id
Tenant_id (deprecated attribute).

propagate_uplink_status
Whether to propagate uplink status of the port. *Type: bool*

qos_policy_id
The ID of the QoS policy attached to the port.

security_group_ids
The IDs of any attached security groups. *Type: list of strs of the security group IDs*

status
The port status. Value is `ACTIVE` or `DOWN`.

trunk_details
Read-only. The trunk referring to this parent port and its subports. Present for trunk parent ports if `trunk-details` extension is loaded. *Type: dict with keys: trunk_id, sub_ports. sub_ports is a list of dicts with keys: port_id, segmentation_type, segmentation_id, mac_address*

trusted

Status of the trusted VIF setting, this value is added to the binding:profile field and passed to services which needs, it, like Nova

updated_at

Timestamp when the port was last updated.

openstack.network.v2.qos_bandwidth_limit_rule

The QoSBandwidthLimitRule Class

The `QoSBandwidthLimitRule` class inherits from *Resource*.

class openstack.network.v2.qos_bandwidth_limit_rule.**QoSBandwidthLimitRule**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'bandwidth_limit_rule'

Singular form of key for resource.

resources_key = 'bandwidth_limit_rules'

Plural form of key for resource.

base_path = '/qos/policies/%(qos_policy_id)s/bandwidth_limit_rules'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

qos_policy_id

The ID of the QoS policy who owns rule.

max_kbps

Maximum bandwidth in kbps.

max_burst_kbps

Maximum burst bandwidth in kbps.

direction

Traffic direction from the tenant point of view (egress, ingress).

openstack.network.v2.qos_dscp_marking_rule

The QoSDSCPMarkingRule Class

The `QoSDSCPMarkingRule` class inherits from *Resource*.

class openstack.network.v2.qos_dscp_marking_rule.**QoSDSCPMarkingRule**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'dscp_marking_rule'

Singular form of key for resource.

resources_key = 'dscp_marking_rules'

Plural form of key for resource.

base_path = '/qos/policies/%(qos_policy_id)s/dscp_marking_rules'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

qos_policy_id

The ID of the QoS policy who owns rule.

dscp_mark

DSCP mark field.

openstack.network.v2.qos_minimum_bandwidth_rule

The QoSMinimumBandwidthRule Class

The QoSMinimumBandwidthRule class inherits from *Resource*.

class openstack.network.v2.qos_minimum_bandwidth_rule.**QoSMinimumBandwidthRule**(*_synchronized=F*
con-
nec-
tion=None,
***at-*
trs)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of self._connection in Resource code should protect itself with a check for None.

resource_key = 'minimum_bandwidth_rule'

Singular form of key for resource.

resources_key = 'minimum_bandwidth_rules'

Plural form of key for resource.

base_path = '/qos/policies/%(qos_policy_id)s/minimum_bandwidth_rules'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

qos_policy_id
The ID of the QoS policy who owns rule.

min_kbps
Minimum bandwidth in kbps.

direction
Traffic direction from the tenant point of view. Valid values: egress

openstack.network.v2.qos_minimum_packet_rate_rule

The QoSMinimumPacketRateRule Class

The `QoSMinimumPacketRateRule` class inherits from *Resource*.

class openstack.network.v2.qos_minimum_packet_rate_rule.**QoSMinimumPacketRateRule**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'minimum_packet_rate_rule'
Singular form of key for resource.

resources_key = 'minimum_packet_rate_rules'
Plural form of key for resource.

base_path = '/qos/policies/%(qos_policy_id)s/minimum_packet_rate_rules'
The base part of the URI for this resource.

allow_create = True
Allow create operation for this resource.

allow_fetch = True
Allow get operation for this resource.

allow_commit = True
Allow update operation for this resource.

allow_delete = True
Allow delete operation for this resource.

allow_list = True
Allow list operation for this resource.

direction

Traffic direction from the tenant point of view. Valid values: (any, egress, ingress)

min_kpps

Minimum packet rate in kpps.

qos_policy_id

The ID of the QoS policy who owns rule.

openstack.network.v2.qos_policy

The QoSPolicy Class

The `QoSPolicy` class inherits from *Resource*.

class openstack.network.v2.qos_policy.**QoSPolicy**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'policy'

Singular form of key for resource.

resources_key = 'policies'

Plural form of key for resource.

base_path = '/qos/policies'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

name

QoS policy name.

project_id

The ID of the project who owns the network. Only administrative users can specify a project ID other than their own.

tenant_id

Tenant_id (deprecated attribute).

description

The QoS policy description.

is_default

Indicates whether this QoS policy is the default policy for this project. *Type: bool*

is_shared

Indicates whether this QoS policy is shared across all projects. *Type: bool*

rules

List of QoS rules applied to this QoS policy.

set_tags(*session*, *tags*)

Sets/Replaces all tags on the resource.

Parameters

- **session** The session to use for making this request.
- **tags** (*list*) List with tags to be set on the resource

openstack.network.v2.qos_rule_type

The QoSRuleType Class

The QoSRuleType class inherits from *Resource*.

class openstack.network.v2.qos_rule_type.**QoSRuleType**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of self._connection in Resource code should protect itself with a check for None.

resource_key = 'rule_type'

Singular form of key for resource.

resources_key = 'rule_types'

Plural form of key for resource.

base_path = '/qos/rule-types'

The base part of the URI for this resource.

allow_create = False

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = False

Allow update operation for this resource.

allow_delete = False

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

type

QoS rule type name.

drivers

List of QoS backend drivers supporting this QoS rule type

openstack.network.v2.quota

The Quota Class

The Quota class inherits from *Resource*.

class openstack.network.v2.quota.**Quota**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of self._connection in Resource code should protect itself with a check for None.

resource_key = 'quota'

Singular form of key for resource.

resources_key = 'quotas'

Plural form of key for resource.

base_path = '/quotas'

The base part of the URI for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

check_limit

Flag to check the quota usage before setting the new limit. *Type: bool*

floating_ips

The maximum amount of floating IPs you can have. *Type: int*

health_monitors

The maximum amount of health monitors you can create. *Type: int*

listeners

The maximum amount of listeners you can create. *Type: int*

load_balancers

The maximum amount of load balancers you can create. *Type: int*

l7_policies

The maximum amount of L7 policies you can create. *Type: int*

networks

The maximum amount of networks you can create. *Type: int*

pools

The maximum amount of pools you can create. *Type: int*

ports

The maximum amount of ports you can create. *Type: int*

project_id

The ID of the project these quota values are for.

rbac_policies

The maximum amount of RBAC policies you can create. *Type: int*

routers

The maximum amount of routers you can create. *Type: int*

subnets

The maximum amount of subnets you can create. *Type: int*

subnet_pools

The maximum amount of subnet pools you can create. *Type: int*

security_group_rules

The maximum amount of security group rules you can create. *Type: int*

security_groups

The maximum amount of security groups you can create. *Type: int*

openstack.network.v2.rbac_policy

The RBACPolicy Class

The RBACPolicy class inherits from *Resource*.

class openstack.network.v2.rbac_policy.**RBACPolicy**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'rbac_policy'

Singular form of key for resource.

resources_key = 'rbac_policies'

Plural form of key for resource.

base_path = '/rbac-policies'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

object_id

ID of the object that this RBAC policy affects.

target_project_id

The ID of the project this RBAC will be enforced.

project_id

The owner project ID.

tenant_id

Tenant_id (deprecated attribute).

object_type

Type of the object that this RBAC policy affects.

action

Action for the RBAC policy.

openstack.network.v2.router

The Router Class

The Router class inherits from *Resource*.

class openstack.network.v2.router.**Router**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of self._connection in Resource code should protect itself with a check for None.

resource_key = 'router'

Singular form of key for resource.

resources_key = 'routers'

Plural form of key for resource.

base_path = '/routers'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

availability_zone_hints

Availability zone hints to use when scheduling the router. *Type: list of availability zone names*

availability_zones

Availability zones for the router. *Type: list of availability zone names*

created_at

Timestamp when the router was created.

description

The router description.

enable_ndp_proxy

The ndp proxy state of the router

external_gateway_info

The `network_id`, for the external gateway. *Type: dict*

flavor_id

The ID of the flavor.

is_admin_state_up

The administrative state of the router, which is up `True` or down `False`. *Type: bool*

is_distributed

The distributed state of the router, which is distributed `True` or not `False`. *Type: bool*

is_ha

The highly-available state of the router, which is highly available `True` or not `False`. *Type: bool*

name

The router name.

project_id

The ID of the project this router is associated with.

tenant_id

Tenant_id (deprecated attribute).

revision_number

Revision number of the router. *Type: int*

routes

The extra routes configuration for the router.

status

The router status.

updated_at

Timestamp when the router was created.

add_interface(*session*, ***body*)

Add an internal interface to a logical router.

Parameters

- **session** (`Adapter`) The session to communicate through.
- **body** (*dict*) The body requested to be updated on the router

Returns

The body of the response as a dictionary.

Raises
SDKException on error.

remove_interface(*session*, ***body*)

Remove an internal interface from a logical router.

Parameters

- **session** (Adapter) The session to communicate through.
- **body** (*dict*) The body requested to be updated on the router

Returns
The body of the response as a dictionary.

Raises
SDKException on error.

add_extra_routes(*session*, *body*)

Add extra routes to a logical router.

Parameters

- **session** (Adapter) The session to communicate through.
- **body** (*dict*) The request body as documented in the api-ref.

Returns
The response as a Router object with the added extra routes.

Raises
SDKException on error.

remove_extra_routes(*session*, *body*)

Remove extra routes from a logical router.

Parameters

- **session** (Adapter) The session to communicate through.
- **body** (*dict*) The request body as documented in the api-ref.

Returns
The response as a Router object with the extra routes left.

Raises
SDKException on error.

add_gateway(*session*, ***body*)

Add an external gateway to a logical router.

Parameters

- **session** (Adapter) The session to communicate through.
- **body** (*dict*) The body requested to be updated on the router

Returns
The body of the response as a dictionary.

remove_gateway(*session*, ***body*)

Remove an external gateway from a logical router.

Parameters

- **session** (Adapter) The session to communicate through.
- **body** (*dict*) The body requested to be updated on the router

Returns

The body of the response as a dictionary.

add_external_gateways(*session*, *body*)

Add external gateways to a router.

Parameters

- **session** (Adapter) The session to communicate through.
- **body** (*dict*) The body requested to be updated on the router

Returns

The body of the response as a dictionary.

update_external_gateways(*session*, *body*)

Update external gateways of a router.

Parameters

- **session** (Adapter) The session to communicate through.
- **body** (*dict*) The body requested to be updated on the router

Returns

The body of the response as a dictionary.

remove_external_gateways(*session*, *body*)

Remove external gateways from a router.

Parameters

- **session** (Adapter) The session to communicate through.
- **body** (*dict*) The body requested to be updated on the router

Returns

The body of the response as a dictionary.

openstack.network.v2.security_group

The SecurityGroup Class

The SecurityGroup class inherits from *Resource*.

class openstack.network.v2.security_group.**SecurityGroup**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.

- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'security_group'

Singular form of key for resource.

resources_key = 'security_groups'

Plural form of key for resource.

base_path = '/security-groups'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

created_at

Timestamp when the security group was created.

description

The security group description.

name

The security group name.

stateful

Whether the security group is stateful or not.

project_id

The ID of the project this security group is associated with.

security_group_rules

A list of *SecurityGroupRule* objects. *Type: list*

tenant_id

The ID of the project this security group is associated with.

updated_at

Timestamp when the security group was last updated.

openstack.network.v2.security_group_rule

The SecurityGroupRule Class

The `SecurityGroupRule` class inherits from *Resource*.

class openstack.network.v2.security_group_rule.**SecurityGroupRule**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'security_group_rule'

Singular form of key for resource.

resources_key = 'security_group_rules'

Plural form of key for resource.

base_path = '/security-group-rules'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = False

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

created_at

Timestamp when the security group rule was created.

description

The security group rule description.

direction

`ingress` or `egress`: The direction in which the security group rule is applied. For a compute instance, an ingress security group rule is applied to incoming ingress traffic for that instance. An egress rule is applied to traffic leaving the instance.

ether_type

Must be IPv4 or IPv6, and addresses represented in CIDR must match the ingress or egress rules.

port_range_max

The maximum port number in the range that is matched by the security group rule. The port_range_min attribute constrains the port_range_max attribute. If the protocol is ICMP, this value must be an ICMP type.

port_range_min

The minimum port number in the range that is matched by the security group rule. If the protocol is TCP or UDP, this value must be less than or equal to the value of the port_range_max attribute. If the protocol is ICMP, this value must be an ICMP type.

project_id

The ID of the project this security group rule is associated with.

protocol

The protocol that is matched by the security group rule. Valid values are `null`, `tcp`, `udp`, and `icmp`.

remote_group_id

The remote security group ID to be associated with this security group rule. You can specify either `remote_group_id` or `remote_address_group_id` or `remote_ip_prefix`.

remote_address_group_id

The remote address group ID to be associated with this security group rule. You can specify either `remote_group_id` or `remote_address_group_id` or `remote_ip_prefix`.

remote_ip_prefix

The remote IP prefix to be associated with this security group rule. You can specify either `remote_group_id` or `remote_address_group_id` or `remote_ip_prefix`. This attribute matches the specified IP prefix as the source or destination IP address of the IP packet depending on direction.

security_group_id

The security group ID to associate with this security group rule.

tenant_id

The ID of the project this security group rule is associated with.

updated_at

Timestamp when the security group rule was last updated.

openstack.network.v2.segment

The Segment Class

The `Segment` class inherits from *Resource*.

`class openstack.network.v2.segment.Segment(`*_synchronized=False*, *connection=None*, ***attrs*`)`

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'segment'

Singular form of key for resource.

resources_key = 'segments'

Plural form of key for resource.

base_path = '/segments'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

description

The segment description.

name

The segment name.

network_id

The ID of the network associated with this segment.

network_type

The type of network associated with this segment, such as `flat`, `geneve`, `gre`, `local`, `vlan` or `vxlan`.

physical_network

The name of the physical network associated with this segment.

segmentation_id

The segmentation ID for this segment. The network type defines the segmentation model, VLAN ID for `vlan` network type and tunnel ID for `geneve`, `gre` and `vxlan` network types.
Type: int

openstack.network.v2.service_profile

The ServiceProfile Class

The `ServiceProfile` class inherits from *Resource*.

class openstack.network.v2.service_profile.**ServiceProfile**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'service_profile'

Singular form of key for resource.

resources_key = 'service_profiles'

Plural form of key for resource.

base_path = '/service_profiles'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

description

Description of the service flavor profile.

driver

Provider driver for the service flavor profile

is_enabled

Sets enabled flag

meta_info

Metainformation of the service flavor profile

project_id

The owner project ID

tenant_id

Tenant_id (deprecated attribute).

openstack.network.v2.service_provider

The Service Provider Class

The `Service Provider` class inherits from *Resource*.

class openstack.network.v2.service_provider.**ServiceProvider**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'service_providers'

Plural form of key for resource.

base_path = '/service-providers'

The base part of the URI for this resource.

allow_create = False

Allow create operation for this resource.

allow_fetch = False

Allow get operation for this resource.

allow_commit = False

Allow update operation for this resource.

allow_delete = False

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

service_type

Service type (FIREWALL, FLAVORS, METERING, QOS, etc..)

name

Name of the service type

is_default

The default value of service type

openstack.network.v2.sfc_flow_classifier

The SfcFlowClassifier Class

The `SfcFlowClassifier` class inherits from *Resource*.

class openstack.network.v2.sfc_flow_classifier.**SfcFlowClassifier**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'flow_classifier'

Singular form of key for resource.

resources_key = 'flow_classifiers'

Plural form of key for resource.

base_path = '/sfc/flow_classifiers'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

description

Human-readable description for the resource.

name

Human-readable name of the resource. Default is an empty string.

protocol

The IP protocol can be represented by a string, an integer, or null. Valid values: any (0), ah (51), dccp (33), egp (8), esp (50), gre (47), icmp (1), icmpv6 (58), igmp (2), ipip (4), ipv6-encap (41), ipv6-frag (44), ipv6-icmp (58), ipv6-nonxt (59), ipv6-opts (60), ipv6-route (43), ospf (89), pgm (113), rsvp (46), sctp (132), tcp (6), udp (17), udplite (136), vrrp (112).

source_port_range_min

Minimum source protocol port.

source_port_range_max

Maximum source protocol port.

destination_port_range_min

Minimum destination protocol port.

destination_port_range_max

Maximum destination protocol port.

source_ip_prefix

The source IP prefix.

destination_ip_prefix

The destination IP prefix.

logical_source_port

The UUID of the source logical port.

logical_destination_port

The UUID of the destination logical port.

l7_parameters

A dictionary of L7 parameters, in the form of logical_source_network: uuid, logical_destination_network: uuid.

summary

Summary field of a Flow Classifier, composed of the protocol, source protcol port, destination ptocolo port, logical_source_port, logical_destination_port and l7_parameters

tenant_id

Tenant_id (deprecated attribute).

openstack.network.v2.sfc_port_chain

The SfcPortChain Class

The `SfcPortChain` class inherits from *Resource*.

class openstack.network.v2.sfc_port_chain.**SfcPortChain**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'port_chain'

Singular form of key for resource.

resources_key = 'port_chains'

Plural form of key for resource.

base_path = '/sfc/port_chains'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

description

Human-readable description for the resource.

name

Human-readable name of the resource. Default is an empty string.

port_pair_groups

List of port-pair-group UUIDs.

flow_classifiers

List of flow-classifier UUIDs.

chain_parameters

A dictionary of chain parameters, correlation values can be mpls and nsh, symmetric can be True or False.

tenant_id

Tenant_id (deprecated attribute).

openstack.network.v2.sfc_port_pair

The SfcPortPair Class

The `SfcPortPair` class inherits from *Resource*.

class openstack.network.v2.sfc_port_pair.**SfcPortPair**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.

- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'port_pair'

Singular form of key for resource.

resources_key = 'port_pairs'

Plural form of key for resource.

base_path = '/sfc/port_pairs'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

description

Human-readable description for the resource.

name

Human-readable name of the resource. Default is an empty string.

ingress

The UUID of the ingress Neutron port.

egress

The UUID of the egress Neutron port.

service_function_parameters

A dictionary of service function parameters, correlation values can be mpls and nsh, weight which can be an int.

tenant_id

Tenant_id (deprecated attribute).

openstack.network.v2.sfc_port_pair_group

The SfcPortPairGroup Class

The `SfcPortPairGroup` class inherits from *Resource*.

class openstack.network.v2.sfc_port_pair_group.**SfcPortPairGroup**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'port_pair_group'

Singular form of key for resource.

resources_key = 'port_pair_groups'

Plural form of key for resource.

base_path = '/sfc/port_pair_groups'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

description

Human-readable description for the resource.

name

Human-readable name of the resource. Default is an empty string.

port_pairs

List of port-pair UUIDs.

port_pair_group_parameters

Dictionary of port pair group parameters, in the form of lb_fields: list of regex (eth|ip|tcp|udp)_(src|dst)), ppg_n_tuple_mapping: ingress_n_tuple or egress_n_tuple. The ingress or egress tuple is a dict with the following keys: source_ip_prefix, destination_ip_prefix, source_port_range_min, source_port_range_max, destination_port_range_min, destination_port_range_max.

is_tap_enabled

True if passive Tap service functions support is enabled, default is False.

tenant_id

Tenant_id (deprecated attribute).

openstack.network.v2.sfc_service_graph

The SfcServiceGraph Class

The `SfcServiceGraph` class inherits from *Resource*.

class openstack.network.v2.sfc_service_graph.**SfcServiceGraph**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'service_graph'

Singular form of key for resource.

resources_key = 'service_graphs'

Plural form of key for resource.

base_path = '/sfc/service_graphs'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

description

Human-readable description for the resource.

name

Human-readable name of the resource. Default is an empty string.

port_chains

A dictionary where the key is the source port chain and the value is a list of destination port chains.

tenant_id

Tenant_id (deprecated attribute).

openstack.network.v2.subnet

The Subnet Class

The Subnet class inherits from *Resource*.

class openstack.network.v2.subnet.**Subnet**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of self._connection in Resource code should protect itself with a check for None.

resource_key = 'subnet'

Singular form of key for resource.

resources_key = 'subnets'

Plural form of key for resource.

base_path = '/subnets'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

allocation_pools

List of allocation pools each of which has a start and an end address for this subnet

cidr

The CIDR.

created_at

Timestamp when the subnet was created.

description

The subnet description.

dns_nameservers

A list of DNS nameservers.

dns_publish_fixed_ip

Whether to publish DNS records for fixed IPs

gateway_ip

The gateway IP address.

host_routes

A list of host routes.

ip_version

The IP version, which is 4 or 6. *Type: int*

ipv6_address_mode

The IPv6 address modes which are dhcpv6-stateful, dhcpv6-stateless or slaac.

ipv6_ra_mode

The IPv6 router advertisements modes which can be slaac, dhcpv6-stateful, dhcpv6-stateless.

is_dhcp_enabled

Set to `True` if DHCP is enabled and `False` if DHCP is disabled. *Type: bool*

name

The subnet name.

network_id

The ID of the attached network.

prefix_length

The prefix length to use for subnet allocation from a subnet pool

project_id

The ID of the project this subnet is associated with.

tenant_id

Tenant_id (deprecated attribute).

segment_id

The ID of the segment this subnet is associated with.

service_types

Service types for this subnet

subnet_pool_id

The subnet pool ID from which to obtain a CIDR.

updated_at

Timestamp when the subnet was last updated.

use_default_subnet_pool

Whether to use the default subnet pool to obtain a CIDR.

openstack.network.v2.subnet_pool

The SubnetPool Class

The SubnetPool class inherits from *Resource*.

class openstack.network.v2.subnet_pool.**SubnetPool**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of self._connection in Resource code should protect itself with a check for None.

resource_key = 'subnetpool'

Singular form of key for resource.

resources_key = 'subnetpools'

Plural form of key for resource.

base_path = '/subnetpools'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

address_scope_id

The ID of the address scope associated with the subnet pool.

created_at

Timestamp when the subnet pool was created.

default_prefix_length

The length of the prefix to allocate when the cidr or prefixlen attributes are omitted when creating a subnet. *Type: int*

default_quota

A per-project quota on the prefix space that can be allocated from the subnet pool for project subnets. For IPv4 subnet pools, default_quota is measured in units of /32. For IPv6 subnet pools, default_quota is measured units of /64. All projects that use the subnet pool have the same prefix quota applied. *Type: int*

description

The subnet pool description.

ip_version

Read-only. The IP address family of the list of prefixes. *Type: int*

is_default

Whether or not this is the default subnet pool. *Type: bool*

is_shared

Indicates whether this subnet pool is shared across all projects. *Type: bool*

maximum_prefix_length

The maximum prefix length that can be allocated from the subnet pool. *Type: int*

minimum_prefix_length

The minimum prefix length that can be allocated from the subnet pool. *Type: int*

name

The subnet pool name.

project_id

The ID of the project that owns the subnet pool.

tenant_id

Tenant_id (deprecated attribute).

prefixes

A list of subnet prefixes that are assigned to the subnet pool. The adjacent prefixes are merged and treated as a single prefix. *Type: list*

revision_number

Revision number of the subnet pool. *Type: int*

updated_at

Timestamp when the subnet pool was last updated.

openstack.network.v2.tap_flow

The TapFlow Class

The `TapFlow` class inherits from *Resource*.

class openstack.network.v2.tap_flow.**TapFlow**(*_synchronized=False*, *connection=None*, ***attrs*)

Tap Flow

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'tap_flow'

Singular form of key for resource.

resources_key = 'tap_flows'

Plural form of key for resource.

base_path = '/taas/tap_flows'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

id

The ID of the tap flow.

name

The tap flows name.

description

The tap flows description.

project_id

The ID of the project that owns the tap flow.

tenant_id

Tenant_id (deprecated attribute).

tap_service_id

The id of the tap_service with which the tap flow is associated

direction

The direction of the tap flow.

status

The status for the tap flow.

source_port

The id of the port the tap flow is associated with

openstack.network.v2.tap_mirror

The TapMirror Class

The `TapMirror` class inherits from *Resource*.

class openstack.network.v2.tap_mirror.**TapMirror**(*_synchronized=False*, *connection=None*, ***attrs*)

Tap Mirror

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'tap_mirror'

Singular form of key for resource.

resources_key = 'tap_mirrors'

Plural form of key for resource.

base_path = '/taas/tap_mirrors'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

id

The ID of the Tap Mirror.

name

The Tap Mirror name.

description

The Tap Mirror description.

project_id

The ID of the project that owns the Tap Mirror.

tenant_id

Tenant_id (deprecated attribute).

port_id

The id of the port the Tap Mirror is associated with

directions

The status for the tap service.

remote_ip

The destination IP address of the Tap Mirror

mirror_type

The type of the Tap Mirror, it can be gre or erspanv1

openstack.network.v2.tap_service

The TapService Class

The `TapService` class inherits from *Resource*.

class openstack.network.v2.tap_service.**TapService**(*_synchronized=False*, *connection=None*, ***attrs*)

Tap Service

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'tap_service'

Singular form of key for resource.

resources_key = 'tap_services'

Plural form of key for resource.

base_path = '/taas/tap_services'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

id

The ID of the tap service.

name

The tap service name.

description

The tap service description.

project_id

The ID of the project that owns the tap service.

tenant_id

Tenant_id (deprecated attribute).

port_id

The id of the port the tap service is associated with

status

The status for the tap service.

VPNaaS Resources

openstack.network.v2.vpn_endpoint_group

The VpnEndpointGroup Class

The VpnEndpointGroup class inherits from *Resource*.

class openstack.network.v2.vpn_endpoint_group.**VpnEndpointGroup**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'endpoint_group'

Singular form of key for resource.

resources_key = 'endpoint_groups'

Plural form of key for resource.

base_path = '/vpn/endpoint-groups'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

description

Human-readable description for the resource.

endpoints

List of endpoints of the same type, for the endpoint group. The values will depend on type.

name

Human-readable name of the resource. Default is an empty string.

tenant_id

Tenant_id (deprecated attribute).

type

The type of the endpoints in the group. A valid value is subnet, cidr, network, router, or vlan. Only subnet and cidr are supported at this moment.

openstack.network.v2.vpn_ike_policy

The VpnIkePolicy Class

The `VpnIkePolicy` class inherits from *Resource*.

class openstack.network.v2.vpn_ike_policy.**VpnIkePolicy**(*_synchronized=False*, *connection=None*, ***attrs*)

VPN IKE policy extension.

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'ikepolicy'

Singular form of key for resource.

resources_key = 'ikepolicies'

Plural form of key for resource.

base_path = '/vpn/ikepolicies'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

ike_version

The IKE version. A valid value is v1 or v2. Default is v1.

name

Human-readable name of the resource. Default is an empty string.

project_id

The ID of the project.

phase1_negotiation_mode

The IKE mode. A valid value is main, which is the default.

openstack.network.v2.vpn_ipsec_policy

The VpnIpsecPolicy Class

The `VpnIpsecPolicy` class inherits from *Resource*.

class openstack.network.v2.vpn_ipsec_policy.**VpnIpsecPolicy**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'ipsecpolicy'

Singular form of key for resource.

resources_key = 'ipsecpolicies'

Plural form of key for resource.

base_path = '/vpn/ipsecpolicies'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

encapsulation_mode

The encapsulation mode. A valid value is tunnel or transport

name

Human-readable name of the resource. Default is an empty string.

project_id

The ID of the project.

phase1_negotiation_mode

The IKE mode. A valid value is main, which is the default.

transform_protocol

The transform protocol. A valid value is ESP, AH, or AH- ESP.

openstack.network.v2.vpn_ipsec_site_connection

The VpnIPSecSiteConnection Class

The `VpnIPSecSiteConnection` class inherits from *Resource*.

class openstack.network.v2.vpn_ipsec_site_connection.**VpnIPSecSiteConnection**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'ipsec_site_connection'

Singular form of key for resource.

resources_key = 'ipsec_site_connections'

Plural form of key for resource.

base_path = '/vpn/ipsec-site-connections'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

dpd

A dictionary with dead peer detection (DPD) protocol controls.

ikepolicy_id

The ID of the IKE policy.

ipsecpolicy_id

The ID of the IPsec policy.

peer_address

The peer gateway public IPv4 or IPv6 address or FQDN.

name

Human-readable name of the resource. Default is an empty string.

project_id

The ID of the project.

psk

The pre-shared key. A valid value is any string.

route_mode

The route mode. A valid value is static, which is the default.

status

The site connection status

vpnservice_id

The ID of the VPN service.

openstack.network.v2.vpn_service

The VpnService Class

The VpnService class inherits from *Resource*.

class openstack.network.v2.vpn_service.**VpnService**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of self._connection in Resource code should protect itself with a check for None.

resource_key = 'vpnservice'

Singular form of key for resource.

resources_key = 'vpnservices'

Plural form of key for resource.

base_path = '/vpn/vpnservices'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

description

Human-readable description for the vpnservice.

external_v4_ip

The external IPv4 address that is used for the VPN service.

external_v6_ip

The external IPv6 address that is used for the VPN service.

is_admin_state_up

The administrative state of the vpnservice, which is up `True` or down `False`. *Type: bool*

name

The vpnservice name.

router_id

ID of the router into which the VPN service is inserted.

project_id

The ID of the project this vpnservice is associated with.

tenant_id

Tenant_id (deprecated attribute).

status

The vpnservice status.

subnet_id

The ID of the subnet on which the tenant wants the vpnservice.

Orchestration Resources

openstack.orchestration.v1.resource

The Resource Class

The `Resource` class inherits from *Resource*.

class openstack.orchestration.v1.resource.**Resource**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'resource'

Singular form of key for resource.

resources_key = 'resources'

Plural form of key for resource.

base_path = '/stacks/%(stack_name)s/%(stack_id)s/resources'

The base part of the URI for this resource.

allow_create = False

Allow create operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_delete = False

Allow delete operation for this resource.

allow_commit = False

Allow update operation for this resource.

links

A list of dictionaries containing links relevant to the resource.

logical_resource_id

ID of the logical resource, usually the literal name of the resource as it appears in the stack template.

name

Name of the resource.

physical_resource_id

ID of the physical resource (if any) that backs up the resource. For example, it contains a nova server ID if the resource is a nova server.

required_by

A list of resource names that depend on this resource. This property facilitates the deduction of resource dependencies. *Type: list*

resource_type

A string representation of the resource type.

status

A string representing the status the resource is currently in.

status_reason

A string that explains why the resource is in its current status.

updated_at

Timestamp of the last update made to the resource.

openstack.orchestration.v1.software_config

The SoftwareConfig Class

The `SoftwareConfig` class inherits from *Resource*.

class openstack.orchestration.v1.software_config.**SoftwareConfig**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'software_config'

Singular form of key for resource.

resources_key = 'software_configs'

Plural form of key for resource.

base_path = '/software_configs'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_commit = False

Allow update operation for this resource.

config

Configuration script or manifest that defines which configuration is performed

created_at

The date and time when the software config resource was created.

group

A string indicating the namespace used for grouping software configs.

inputs

A list of schemas each representing an input this software config expects.

name

Name of the software config.

options

A string that contains options that are specific to the configuration management tool that this resource uses.

outputs

A list of schemas each representing an output this software config produces.

create(*session*, *prepend_key=False*, **args*, ***kwargs*)

Create a remote resource based on this instance.

Parameters

- **session** (Adapter) The session to use for making this request.
- **prepend_key** A boolean indicating whether the resource_key should be prepended in a resource creation request. Default to True.
- **base_path** (*str*) Base part of the URI for creating resources, if different from *base_path*.
- **resource_request_key** (*str*) Overrides the usage of self.resource_key when prepending a key to the request body. Ignored if *prepend_key* is false.
- **resource_response_key** (*str*) Overrides the usage of self.resource_key when processing response bodies. Ignored if *prepend_key* is false.
- **microversion** (*str*) API version to override the negotiated one.
- **params** (*dict*) Additional params to pass.

Returns

This Resource instance.

Raises

MethodNotSupported if Resource.allow_create is not set to True.

openstack.orchestration.v1.software_deployment

The SoftwareDeployment Class

The SoftwareDeployment class inherits from *Resource*.

class openstack.orchestration.v1.software_deployment.**SoftwareDeployment**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of self._connection in Resource code should protect itself with a check for None.

resource_key = 'software_deployment'

Singular form of key for resource.

resources_key = 'software_deployments'

Plural form of key for resource.

base_path = '/software_deployments'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_commit = True

Allow update operation for this resource.

action

The stack action that triggers this deployment resource.

config_id

The UUID of the software config resource that runs when applying to the server.

input_values

A map containing the names and values of all inputs to the config.

output_values

A map containing the names and values from the deployment.

server_id

The UUID of the compute server to which the configuration applies.

stack_user_project_id

The ID of the authentication project which can also perform operations on this deployment.

status

Current status of the software deployment.

status_reason

Error description for the last status change.

created_at

The date and time when the software deployment resource was created.

updated_at

The date and time when the software deployment resource was created.

create(*session*, *prepend_key=False*, **args*, ***kwargs*)

Create a remote resource based on this instance.

Parameters

- **session** (Adapter) The session to use for making this request.
- **prepend_key** A boolean indicating whether the resource_key should be prepended in a resource creation request. Default to True.

- **base_path** (*str*) Base part of the URI for creating resources, if different from *base_path*.
- **resource_request_key** (*str*) Overrides the usage of self.resource_key when prepending a key to the request body. Ignored if *prepend_key* is false.
- **resource_response_key** (*str*) Overrides the usage of self.resource_key when processing response bodies. Ignored if *prepend_key* is false.
- **microversion** (*str*) API version to override the negotiated one.
- **params** (*dict*) Additional params to pass.

Returns
This Resource instance.

Raises
MethodNotSupported if Resource.allow_create is not set to True.

commit(*session*, *prepend_key=False*, **args*, ***kwargs*)

Commit the state of the instance to the remote resource.

Parameters

- **session** (Adapter) The session to use for making this request.
- **prepend_key** A boolean indicating whether the resource_key should be prepended in a resource update request. Default to True.
- **retry_on_conflict** (*bool*) Whether to enable retries on HTTP CONFLICT (409). Value of None leaves the *Adapter* defaults.
- **base_path** (*str*) Base part of the URI for modifying resources, if different from *base_path*.
- **microversion** (*str*) API version to override the negotiated one.
- **kwargs** (*dict*) Parameters that will be passed to _prepare_request()

Returns
This Resource instance.

Raises
MethodNotSupported if Resource.allow_commit is not set to True.

openstack.orchestration.v1.stack

The Stack Class

The Stack class inherits from *Resource*.

class openstack.orchestration.v1.stack.**Stack**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.

- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'stack'

Singular form of key for resource.

resources_key = 'stacks'

Plural form of key for resource.

base_path = '/stacks'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

capabilities

Placeholder for AWS compatible template listing capabilities required by the stack.

created_at

Timestamp of the stack creation.

description

A text description of the stack.

deleted

A list of resource objects that will be deleted if a stack update is performed.

deleted_at

Timestamp of the stack deletion.

environment

A JSON environment for the stack.

environment_files

An ordered list of names for environment files found in the files dict.

files

Additional files referenced in the template or the environment

files_container

Name of the container in swift that has child templates and environment files.

is_rollback_disabled

Whether the stack will support a rollback operation on stack create/update failures. *Type: bool*

links

A list of dictionaries containing links relevant to the stack.

name

Name of the stack.

notification_topics

Placeholder for future extensions where stack related events can be published.

outputs

A list containing output keys and values from the stack, if any.

owner_id

The ID of the owner stack if any.

parameters

A dictionary containing the parameter names and values for the stack.

parent_id

The ID of the parent stack if any

replaced

A list of resource objects that will be replaced if a stack update is performed.

status

A string representation of the stack status, e.g. `CREATE_COMPLETE`.

status_reason

A text explaining how the stack transits to its current status.

tags

A list of strings used as tags on the stack

template

A dict containing the template use for stack creation.

template_description

Stack template description text. Currently contains the same text as that of the `description` property.

template_url

A string containing the URL where a stack template can be found.

timeout_mins

Stack operation timeout in minutes.

unchanged

A list of resource objects that will remain unchanged if a stack update is performed.

updated

A list of resource objects that will have their properties updated in place if a stack update is performed.

updated_at

Timestamp of last update on the stack.

user_project_id

The ID of the user project created for this stack.

create(*session*, *prepend_key=False*, **args*, ***kwargs*)

Create a remote resource based on this instance.

Parameters

- **session** (Adapter) The session to use for making this request.
- **prepend_key** A boolean indicating whether the resource_key should be prepended in a resource creation request. Default to True.
- **base_path** (*str*) Base part of the URI for creating resources, if different from *base_path*.
- **resource_request_key** (*str*) Overrides the usage of self.resource_key when prepending a key to the request body. Ignored if *prepend_key* is false.
- **resource_response_key** (*str*) Overrides the usage of self.resource_key when processing response bodies. Ignored if *prepend_key* is false.
- **microversion** (*str*) API version to override the negotiated one.
- **params** (*dict*) Additional params to pass.

Returns

This Resource instance.

Raises

MethodNotSupported if Resource.allow_create is not set to True.

commit(*session*, *prepend_key=True*, *has_body=True*, *retry_on_conflict=None*, *base_path=None*, ***, *microversion=None*, *preview=False*, ***kwargs*)

Commit the state of the instance to the remote resource.

Parameters

- **session** (Adapter) The session to use for making this request.
- **prepend_key** A boolean indicating whether the resource_key should be prepended in a resource update request. Default to True.
- **retry_on_conflict** (*bool*) Whether to enable retries on HTTP CONFLICT (409). Value of None leaves the *Adapter* defaults.
- **base_path** (*str*) Base part of the URI for modifying resources, if different from *base_path*.
- **microversion** (*str*) API version to override the negotiated one.
- **kwargs** (*dict*) Parameters that will be passed to _prepare_request()

Returns

This Resource instance.

Raises

MethodNotSupported if Resource.allow_commit is not set to True.

export(*session*)

Export a stack data

Parameters
session The session to use for making this request.

Returns
A dictionary containing the stack data.

suspend(*session*)

Suspend a stack

Parameters
session The session to use for making this request

Returns
None

resume(*session*)

Resume a stack

Parameters
session The session to use for making this request

Returns
None

fetch(*session*, *requires_id=True*, *base_path=None*, *error_message=None*, *skip_cache=False*, *, *resolve_outputs=True*, ***params*)

Get a remote resource based on this instance.

Parameters

- **session** (Adapter) The session to use for making this request.
- **requires_id** (*boolean*) A boolean indicating whether resource ID should be part of the requested URI.
- **base_path** (*str*) Base part of the URI for fetching resources, if different from *base_path*.
- **error_message** (*str*) An Error message to be returned if requested object does not exist.
- **skip_cache** (*bool*) A boolean indicating whether optional API cache should be skipped for this invocation.
- **resource_response_key** (*str*) Overrides the usage of self.resource_key when processing the response body.
- **microversion** (*str*) API version to override the negotiated one.
- **params** (*dict*) Additional parameters that can be consumed.

Returns
This Resource instance.

Raises
MethodNotSupported if Resource.allow_fetch is not set to True.

Raises

NotFoundException if the resource was not found.

classmethod find(*session*, *name_or_id*, *ignore_missing=True*, ***params*)

Find a resource by its name or id.

Parameters

- **session** (Adapter) The session to use for making this request.
- **name_or_id** This resources identifier, if needed by the request. The default is None.
- **ignore_missing** (*bool*) When set to False *NotFoundException* will be raised when the resource does not exist. When set to True, None will be returned when attempting to find a nonexistent resource.
- **params** (*dict*) Any additional parameters to be passed into underlying methods, such as to *existing()* in order to pass on URI parameters.

Returns

The Resource object matching the given name or id or None if nothing matches.

Raises

openstack.exceptions.DuplicateResource if more than one resource is found for this request.

Raises

openstack.exceptions.NotFoundException if nothing is found and ignore_missing is False.

openstack.orchestration.v1.stack_environment

The StackEnvironment Class

The StackEnvironment class inherits from *Resource*.

class openstack.orchestration.v1.stack_environment.**StackEnvironment**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of self._connection in Resource code should protect itself with a check for None.

base_path = '/stacks/%(stack_name)s/%(stack_id)s/environment'

The base part of the URI for this resource.

allow_create = False

Allow create operation for this resource.

allow_list = False

Allow list operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_delete = False

Allow delete operation for this resource.

allow_commit = False

Allow update operation for this resource.

name

Name of the stack where the template is referenced.

id

ID of the stack where the template is referenced.

encrypted_param_names

A list of parameter names whose values are encrypted

event_sinks

A list of event sinks

parameter_defaults

A map of parameters and their default values defined for the stack.

parameters

A map of parametes defined in the stack template.

resource_registry

A map containing customized resource definitions.

openstack.orchestration.v1.stack_event

The StackEvent Class

The `StackEvent` class inherits from *Resource*.

class openstack.orchestration.v1.stack_event.**StackEvent**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

base_path = '/stacks/%(stack_name)s/%(stack_id)s/events'

The base part of the URI for this resource.

resources_key = 'events'

Plural form of key for resource.

allow_create = False

Allow create operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_delete = False

Allow delete operation for this resource.

allow_commit = False

Allow update operation for this resource.

event_time

The date and time when the event was created

id

The ID of the event object

links

A list of dictionaries containing links relevant to the stack.

logical_resource_id

The ID of the logical stack resource.

physical_resource_id

The ID of the stack physical resource.

resource_name

The name of the resource.

resource_status

The status of the resource.

resource_status_reason

The reason for the current stack resource state.

openstack.orchestration.v1.stack_files

The StackFiles Class

The `StackFiles` class inherits from *Resource*.

class openstack.orchestration.v1.stack_files.**StackFiles**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

base_path = '/stacks/%(stack_name)s/%(stack_id)s/files'

The base part of the URI for this resource.

allow_create = False

Allow create operation for this resource.

allow_list = False

Allow list operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_delete = False

Allow delete operation for this resource.

allow_commit = False

Allow update operation for this resource.

name

Name of the stack where the template is referenced.

id

ID of the stack where the template is referenced.

fetch(*session*, *requires_id=False*, *base_path=None*, **args*, ***kwargs*)

Get a remote resource based on this instance.

Parameters

- **session** (Adapter) The session to use for making this request.
- **requires_id** (*boolean*) A boolean indicating whether resource ID should be part of the requested URI.
- **base_path** (*str*) Base part of the URI for fetching resources, if different from *base_path*.
- **error_message** (*str*) An Error message to be returned if requested object does not exist.
- **skip_cache** (*bool*) A boolean indicating whether optional API cache should be skipped for this invocation.
- **resource_response_key** (*str*) Overrides the usage of self.resource_key when processing the response body.
- **microversion** (*str*) API version to override the negotiated one.
- **params** (*dict*) Additional parameters that can be consumed.

Returns

This Resource instance.

Raises

MethodNotSupported if `Resource.allow_fetch` is not set to `True`.

Raises

NotFoundException if the resource was not found.

openstack.orchestration.v1.stack_template

The StackTemplate Class

The `StackTemplate` class inherits from *Resource*.

class openstack.orchestration.v1.stack_template.**StackTemplate**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

base_path = '/stacks/%(stack_name)s/%(stack_id)s/template'

The base part of the URI for this resource.

allow_create = False

Allow create operation for this resource.

allow_list = False

Allow list operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_delete = False

Allow delete operation for this resource.

allow_commit = False

Allow update operation for this resource.

name

Name of the stack where the template is referenced.

stack_id

ID of the stack where the template is referenced.

description

The description specified in the template

heat_template_version

The version of the orchestration HOT template.

outputs

Key and value that contain output data.

parameters

Key and value pairs that contain template parameters

resources

Key and value pairs that contain definition of resources in the template

openstack.orchestration.v1.template

The Template Class

The `Template` class inherits from *Resource*.

class openstack.orchestration.v1.template.**Template**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

allow_create = False

Allow create operation for this resource.

allow_list = False

Allow list operation for this resource.

allow_fetch = False

Allow get operation for this resource.

allow_delete = False

Allow delete operation for this resource.

allow_commit = False

Allow update operation for this resource.

description

The description specified in the template

parameters

Key and value pairs that contain template parameters

parameter_groups

A list of parameter groups each contains a lsit of parameter names.

Object Store Resources

openstack.object_store.v1.account

The Account Class

The Account class inherits from *Resource*.

class openstack.object_store.v1.account.**Account**(*metadata=None*, ***attrs*)

Process and save metadata known at creation stage

base_path = '/'

The base part of the URI for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_head = True

Allow head operation for this resource.

account_bytes_used

The total number of bytes that are stored in Object Storage for the account.

account_container_count

The number of containers.

account_object_count

The number of objects in the account.

meta_temp_url_key

The secret key value for temporary URLs. If not set, this header is not returned by this operation.

meta_temp_url_key_2

A second secret key value for temporary URLs. If not set, this header is not returned by this operation.

timestamp

The timestamp of the transaction.

has_body = False

Do responses for this resource have bodies

requires_id = False

Do calls for this resource require an id

set_temp_url_key(*proxy*, *key*, *secondary=False*)

Set the temporary url key for the account.

Parameters

- **proxy** (*Proxy*) The proxy to use for making this request.
- **key** Text of the key to use.

- **secondary** (*bool*) Whether this should set the secondary key. (defaults to False)

openstack.object_store.v1.container

The Container Class

The Container class inherits from *Resource*.

class openstack.object_store.v1.container.**Container**(*metadata=None*, ***attrs*)

Process and save metadata known at creation stage

base_path = '/'

The base part of the URI for this resource.

pagination_key = 'X-Account-Container-Count'

Key used for pagination links

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_head = True

Allow head operation for this resource.

name

The name of the container.

count

The number of objects in the container.

bytes

The total number of bytes that are stored in Object Storage for the container.

object_count

The number of objects.

bytes_used

The count of bytes used in total.

timestamp

The timestamp of the transaction.

is_newest

If set to True, Object Storage queries all replicas to return the most recent one. If you omit this header, Object Storage responds faster after it finds one valid replica. Because setting this header to True is more expensive for the back end, use it only when it is absolutely needed. *Type: bool*

read_ACL

The ACL that grants read access. If not set, this header is not returned by this operation.

write_ACL

The ACL that grants write access. If not set, this header is not returned by this operation.

sync_to

The destination for container synchronization. If not set, this header is not returned by this operation.

sync_key

The secret key for container synchronization. If not set, this header is not returned by this operation.

versions_location

Enables versioning on this container. The value is the name of another container. You must UTF-8-encode and then URL-encode the name before you include it in the header. To disable versioning, set the header to an empty string.

history_location

Enables versioning on the container.

content_type

The MIME type of the list of names.

is_content_type_detected

If set to true, Object Storage guesses the content type based on the file extension and ignores the value sent in the Content-Type header, if present. *Type: bool*

storage_policy

Storage policy used by the container. It is not possible to change policy of an existing container

if_none_match

In combination with Expect: 100-Continue, specify an If-None-Match: * header to query whether the server already has a copy of the object before any data is sent.

meta_temp_url_key

The secret key value for temporary URLs. If not set, this header is not returned by this operation.

meta_temp_url_key_2

A second secret key value for temporary URLs. If not set, this header is not returned by this operation.

classmethod new(***kwargs*)

Create a new instance of this resource.

When creating the instance set the `_synchronized` parameter of `Resource` to `False` to indicate that the resource does not yet exist on the server side. This marks all attributes

passed in **kwargs as dirty on the resource, and thusly tracked as necessary in subsequent calls such as update().

Parameters

kwargs (*dict*) Each of the named arguments will be set as attributes on the resulting Resource object.

create(*session*, *prepend_key=True*, *base_path=None*, ***kwargs*)

Create a remote resource based on this instance.

Parameters

- **session** (Adapter) The session to use for making this request.
- **prepend_key** A boolean indicating whether the resource_key should be prepended in a resource creation request. Default to True.

Returns

This Resource instance.

Raises

MethodNotSupported if Resource.allow_create is not set to True.

set_temp_url_key(*proxy*, *key*, *secondary=False*)

Set the temporary url key for a container.

Parameters

- **proxy** (*Proxy*) The proxy to use for making this request.
- **container** The value can be the name of a container or a *Container* instance.
- **key** Text of the key to use.
- **secondary** (*bool*) Whether this should set the second key. (defaults to False)

openstack.object_store.v1.obj

The Object Class

The Object class inherits from *Resource*.

class openstack.object_store.v1.obj.**Object**(*data=None*, ***attrs*)

Process and save metadata known at creation stage

base_path = '/%(container)s'

The base part of the URI for this resource.

pagination_key = 'X-Container-Object-Count'

Key used for pagination links

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_head = True

Allow head operation for this resource.

container

The unique name for the container.

name

The unique name for the object.

is_newest

If set to True, Object Storage queries all replicas to return the most recent one. If you omit this header, Object Storage responds faster after it finds one valid replica. Because setting this header to True is more expensive for the back end, use it only when it is absolutely needed. *Type: bool*

range

TODO(briancurtin) theres a lot of content here

if_match

See http://www.ietf.org/rfc/rfc2616.txt.

if_none_match

In combination with Expect: 100-Continue, specify an If-None-Match: * header to query whether the server already has a copy of the object before any data is sent.

if_modified_since

See http://www.ietf.org/rfc/rfc2616.txt.

if_unmodified_since

See http://www.ietf.org/rfc/rfc2616.txt.

signature

Used with temporary URLs to sign the request. For more information about temporary URLs, see OpenStack Object Storage API v1 Reference.

expires_at

Used with temporary URLs to specify the expiry time of the signature. For more information about temporary URLs, see OpenStack Object Storage API v1 Reference.

manifest

If present, this is a dynamic large object manifest object. The value is the container and object name prefix of the segment objects in the form container/prefix.

multipart_manifest

If you include the multipart-manifest=get query parameter and the object is a large object, the object contents are not returned. Instead, the manifest is returned in the X-Object-Manifest response header for dynamic large objects or in the response body for static large objects.

content_length

HEAD operations do not return content. However, in this operation the value in the Content-Length header is not the size of the response body. Instead it contains the size of the object, in bytes.

content_type

The MIME type of the object.

accept_ranges

The type of ranges that the object accepts.

etag

For objects smaller than 5 GB, this value is the MD5 checksum of the object content. The value is not quoted. For manifest objects, this value is the MD5 checksum of the concatenated string of MD5 checksums and ETags for each of the segments in the manifest, and not the MD5 checksum of the content that was downloaded. Also the value is enclosed in double-quote characters. You are strongly recommended to compute the MD5 checksum of the response body as it is received and compare this value with the one in the ETag header. If they differ, the content was corrupted, so retry the operation.

is_static_large_object

Set to True if this object is a static large object manifest object. *Type: bool*

content_encoding

If set, the value of the Content-Encoding metadata. If not set, this header is not returned by this operation.

content_disposition

If set, specifies the override behavior for the browser. For example, this header might specify that the browser use a download program to save this file rather than show the file, which is the default. If not set, this header is not returned by this operation.

delete_after

Specifies the number of seconds after which the object is removed. Internally, the Object Storage system stores this value in the X-Delete-At metadata item.

delete_at

If set, the time when the object will be deleted by the system in the format of a UNIX Epoch timestamp. If not set, this header is not returned by this operation.

object_manifest

If set, to this is a dynamic large object manifest object. The value is the container and object name prefix of the segment objects in the form container/prefix.

timestamp

The timestamp of the transaction.

last_modified_at

The date and time that the object was created or the last time that the metadata was changed.

transfer_encoding

Set to chunked to enable chunked transfer encoding. If used, do not set the Content-Length header to a non-zero value.

is_content_type_detected

If set to true, Object Storage guesses the content type based on the file extension and ignores the value sent in the Content-Type header, if present. *Type: bool*

copy_from

If set, this is the name of an object used to create the new object by copying the X-Copy-From object. The value is in form {container}/{object}. You must UTF-8-encode and then URL-encode the names of the container and object before you include them in the header. Using PUT with X-Copy-From has the same effect as using the COPY operation to copy an object.

symlink_target

If present, this is a symlink object. The value is the relative path of the target object in the format <container>/<object>.

symlink_target_account

If present, and X-Symlink-Target is present, then this is a cross-account symlink to an object in the account specified in the value.

access_control_allow_origin

CORS for RAX (deviating from standard)

has_body = False

Do responses for this resource have bodies

create(*session*, *prepend_key=True*, *base_path=None*, ***kwargs*)

Create a remote resource based on this instance.

Parameters

- **session** (Adapter) The session to use for making this request.
- **prepend_key** A boolean indicating whether the resource_key should be prepended in a resource creation request. Default to True.
- **base_path** (*str*) Base part of the URI for creating resources, if different from *base_path*.
- **resource_request_key** (*str*) Overrides the usage of self.resource_key when prepending a key to the request body. Ignored if *prepend_key* is false.
- **resource_response_key** (*str*) Overrides the usage of self.resource_key when processing response bodies. Ignored if *prepend_key* is false.
- **microversion** (*str*) API version to override the negotiated one.
- **params** (*dict*) Additional params to pass.

Returns

This Resource instance.

Raises

MethodNotSupported if Resource.allow_create is not set to True.

Placement v1 Resources

openstack.placement.v1.resource_class

The ResourceClass Class

The `ResourceClass` class inherits from *Resource*.

class openstack.placement.v1.resource_class.**ResourceClass**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = None

Singular form of key for resource.

resources_key = 'resource_classes'

Plural form of key for resource.

base_path = '/resource_classes'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

name

The name of this resource.

openstack.placement.v1.resource_provider

The ResourceProvider Class

The `ResourceProvider` class inherits from *Resource*.

class openstack.placement.v1.resource_provider.**ResourceProvider**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = None

Singular form of key for resource.

resources_key = 'resource_providers'

Plural form of key for resource.

base_path = '/resource_providers'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

aggregates

Aggregates

id

The UUID of a resource provider.

generation

A consistent view marker that assists with the management of concurrent resource provider updates.

links

Links pertaining to this flavor. This is a list of dictionaries, each including keys `href` and `rel`.

name

The name of this resource provider.

parent_provider_id

The UUID of the immediate parent of the resource provider.

root_provider_id

Read-only UUID of the top-most provider in this provider tree.

fetch_aggregates(*session*)

List aggregates set on the resource provider

Parameters

session The session to use for making this request

Returns

The resource provider with aggregates populated

set_aggregates(*session*, *aggregates=None*)

Replaces aggregates on the resource provider

Parameters

- **session** The session to use for making this request
- **aggregates** (*list*) List of aggregates

Returns

The resource provider with updated aggregates populated

openstack.placement.v1.resource_provider_inventory

The ResourceProviderInventory Class

The ResourceProviderInventory class inherits from *Resource*.

class openstack.placement.v1.resource_provider_inventory.**ResourceProviderInventory**(*_synchroni con- nec- tion=None **at- trs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of self._connection in Resource code should protect itself with a check for None.

resource_key = None

Singular form of key for resource.

resources_key = None

Plural form of key for resource.

base_path = '/resource_providers/%(resource_provider_id)s/inventories'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

resource_provider_id

The UUID of a resource provider.

resource_class

The name of the resource class.

resource_provider_generation

A consistent view marker that assists with the management of concurrent resource provider updates.

allocation_ratio

It is used in determining whether consumption of the resource of the provider can exceed physical constraints.

max_unit

A maximum amount any single allocation against an inventory can have.

min_unit

A minimum amount any single allocation against an inventory can have.

reserved

The amount of the resource a provider has reserved for its own use.

step_size

A representation of the divisible amount of the resource that may be requested. For example, step_size = 5 means that only values divisible by 5 (5, 10, 15, etc.) can be requested.

total

The actual amount of the resource that the provider can accommodate.

commit(*session*, *prepend_key=True*, *has_body=True*, *retry_on_conflict=None*, *base_path=None*, ***, *microversion=None*, ***kwargs*)

Commit the state of the instance to the remote resource.

Parameters

- **session** (Adapter) The session to use for making this request.

- **prepend_key** A boolean indicating whether the resource_key should be prepended in a resource update request. Default to True.
- **retry_on_conflict** (*bool*) Whether to enable retries on HTTP CONFLICT (409). Value of `None` leaves the *Adapter* defaults.
- **base_path** (*str*) Base part of the URI for modifying resources, if different from *base_path*.
- **microversion** (*str*) API version to override the negotiated one.
- **kwargs** (*dict*) Parameters that will be passed to _prepare_request()

Returns
This `Resource` instance.

Raises
MethodNotSupported if `Resource.allow_commit` is not set to `True`.

classmethod list(*session*, *paginated=True*, *base_path=None*, *allow_unknown_params=False*, ***, *microversion=None*, ***params*)

This method is a generator which yields resource objects.

A re-implementation of *list()* that handles placements single, unpaginated list implementation.

Refer to *list()* for full documentation including parameter, exception and return type documentation.

openstack.placement.v1.trait

The Trait Class

The `Trait` class inherits from *Resource*.

class openstack.placement.v1.trait.**Trait**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = None

Singular form of key for resource.

resources_key = None

Plural form of key for resource.

base_path = '/traits'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

create_method = 'PUT'

Method for creating a resource (POST, PUT)

name

The name of this resource.

classmethod list(*session*, *paginated=True*, *base_path=None*, *allow_unknown_params=False*, ***, *microversion=None*, ***params*)

This method is a generator which yields resource objects.

A re-implementation of *list()* that handles the list of strings (as opposed to a list of objects) that this call returns.

Refer to *list()* for full documentation including parameter, exception and return type documentation.

Shared File System service resources

openstack.shared_file_system.v2.availability_zone

The AvailabilityZone Class

The `AvailabilityZone` class inherits from *Resource*.

class openstack.shared_file_system.v2.availability_zone.**AvailabilityZone**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'availability_zone'

Singular form of key for resource.

resources_key = 'availability_zones'

Plural form of key for resource.

base_path = '/availability-zones'

The base part of the URI for this resource.

allow_create = False

Allow create operation for this resource.

allow_fetch = False

Allow get operation for this resource.

allow_commit = False

Allow update operation for this resource.

allow_delete = False

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

id

Properties The ID of the availability zone

name

The name of the availability zone.

created_at

Date and time the availability zone was created at.

updated_at

Date and time the availability zone was last updated at.

openstack.shared_file_system.v2.storage_pool

The StoragePool Class

The `StoragePool` class inherits from *Resource*.

class openstack.shared_file_system.v2.storage_pool.**StoragePool**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'pools'

Plural form of key for resource.

base_path = '/scheduler-stats/pools'

The base part of the URI for this resource.

allow_create = False

Allow create operation for this resource.

allow_fetch = False

Allow get operation for this resource.

allow_commit = False

Allow update operation for this resource.

allow_delete = False

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_head = False

Allow head operation for this resource.

backend

Properties The name of the back end.

host

The host of the back end.

pool

The pool for the back end

capabilities

The back end capabilities.

openstack.shared_file_system.v2.limit

The Limit Class

The `Limit` class inherits from *Resource*.

class openstack.shared_file_system.v2.limit.**Limit**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resources_key = 'limits'

Plural form of key for resource.

base_path = '/limits'

The base part of the URI for this resource.

allow_create = False

Allow create operation for this resource.

allow_fetch = False

Allow get operation for this resource.

allow_commit = False

Allow update operation for this resource.

allow_delete = False

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_head = False

Allow head operation for this resource.

maxTotalReplicaGigabytes

Properties The maximum number of replica gigabytes that are allowed in a project.

maxTotalShares

The total maximum number of shares that are allowed in a project.

maxTotalShareGigabytes

The total maximum number of share gigabytes that are allowed in a project.

maxTotalShareNetworks

The total maximum number of share-networks that are allowed in a project.

maxTotalShareSnapshots

The total maximum number of share snapshots that are allowed in a project.

maxTotalShareReplicas

The maximum number of share replicas that is allowed.

maxTotalSnapshotGigabytes

The total maximum number of snapshot gigabytes that are allowed in a project.

totalReplicaGigabytesUsed

The total number of replica gigabytes used in a project by share replicas.

totalShareGigabytesUsed

The total number of gigabytes used in a project by shares.

totalSharesUsed

The total number of created shares in a project.

totalShareNetworksUsed

The total number of created share-networks in a project.

totalShareSnapshotsUsed

The total number of created share snapshots in a project.

totalSnapshotGigabytesUsed

The total number of gigabytes used in a project by snapshots.

totalShareReplicasUsed

The total number of created share replicas in a project.

openstack.shared_file_system.v2.share

The Share Class

The Share class inherits from *Resource*.

class openstack.shared_file_system.v2.share.**Share**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'share'

Singular form of key for resource.

resources_key = 'shares'

Plural form of key for resource.

base_path = '/shares'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_head = False

Allow head operation for this resource.

allow_delete = True

Allow delete operation for this resource.

access_rules_status

Properties The share instance access rules status. A valid value is active, error, or syncing.

availability_zone
: The availability zone.

created_at
: The date and time stamp when the resource was created within the services database.

description
: The user defined description of the resource.

host
: The share host name.

is_public
: The level of visibility for the share.

is_creating_new_share_from_snapshot_supported
: Whether or not this share supports snapshots that can be cloned into new shares.

is_mounting_snapshot_supported
: Whether the shares snapshots can be mounted directly and access controlled independently or not.

is_reverting_to_snapshot_supported
: Whether the share can be reverted to its latest snapshot or not.

is_snapshot_supported
: An extra specification that filters back ends by whether the share supports snapshots or not.

is_replicated
: Indicates whether the share has replicas or not.

metadata
: One or more metadata key and value pairs as a dictionary of strings.

progress
: The progress of the share creation.

project_id
: The ID of the project that owns the resource.

replication_type
: The share replication type. Valid values are none, readable, writable and dr.

share_group_id
: The UUID of the share group that this shares belongs to.

share_network_id
: The share network ID.

share_protocol
: The Shared File Systems protocol. A valid value is NFS, CIFS, GlusterFS, HDFS, CephFS, MAPRFS

share_server_id
: The UUID of the share server.

share_type

The UUID of the share type. In minor versions, this parameter is a share type name, as a string.

share_type_name

Name of the share type.

size

The share size, in GiBs.

snapshot_id

The UUID of the snapshot that was used to create the share.

source_share_group_snapshot_member_id

The ID of the group snapshot instance that was used to create this share.

status

The share status

task_state

For the share migration, the migration task state.

user_id

ID of the user that the share was created by.

display_name

Display name for updating name

display_description

Display description for updating description

extend_share(*session*, *new_size*, *force=False*)

Extend the share size.

Parameters

- **new_size** (*float*) The new size of the share in GiB.
- **force** (*bool*) Whether or not to use force, bypassing the scheduler. Requires admin privileges. Defaults to False.

Returns

The result of the action.

Return type

None

shrink_share(*session*, *new_size*)

Shrink the share size.

Parameters

new_size (*float*) The new size of the share in GiB.

Returns

None

revert_to_snapshot(*session*, *snapshot_id*)

Revert the share to the given snapshot.

Parameters

snapshot_id (*str*) The id of the snapshot to revert to.

Returns

None

manage(*session*, *protocol*, *export_path*, *service_host*, ***params*)

Manage a share.

Parameters

- **session** A session object used for sending request.
- **protocol** (*str*) The shared file systems protocol of this share.
- **export_path** (*str*) The export path formatted according to the protocol.
- **service_host** (*str*) The manage-share service host.
- **params** (*kwargs*) Optional parameters to be sent. Available parameters include:
  - name: The user defined name of the resource.
  - share_type: The name or ID of the share type to be used to create the resource.
  - driver_options: A set of one or more key and value pairs, as a dictionary of strings, that describe driver options.
  - is_public: The level of visibility for the share.
  - description: The user defiend description of the resource.
  - share_server_id: The UUID of the share server.

Returns

The share that was managed.

unmanage(*session*)

Unmanage a share.

Parameters

session A session object used for sending request.

Returns

None

openstack.shared_file_system.v2.share_instance

The ShareInstance Class

The ShareInstance class inherits from *Resource*.

class openstack.shared_file_system.v2.share_instance.**ShareInstance**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'share_instance'

Singular form of key for resource.

resources_key = 'share_instances'

Plural form of key for resource.

base_path = '/share_instances'

The base part of the URI for this resource.

allow_create = False

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = False

Allow update operation for this resource.

allow_delete = False

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_head = False

Allow head operation for this resource.

access_rules_status

Properties The share instance access rules status. A valid value is active, error, or syncing.

availability_zone

The name of the availability zone the share exists within.

cast_rules_to_readonly

If the share instance has its cast_rules_to_readonly attribute set to True, all existing access rules be cast to read/only.

created_at

The date and time stamp when the resource was created within the services database.

host

The host name of the service back end that the resource is contained within.

progress

The progress of the share creation.

replica_state

The share replica state. Has set value only when replication is used. List of possible values: active, in_sync, out_of_sync, error

share_id

The UUID of the share to which the share instance belongs to.

share_network_id

The share network ID where the resource is exported to.

share_server_id

The UUID of the share server.

status

The share or share instance status.

reset_status(*session*, *reset_status*)

Reset share instance to given status

force_delete(*session*)

Force delete share instance

openstack.shared_file_system.v2.share_network_subnet

The ShareNetworkSubnet Class

The `ShareNetworkSubnet` class inherits from *Resource*.

class openstack.shared_file_system.v2.share_network_subnet.**ShareNetworkSubnet**(*_synchronized=F*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'share_network_subnet'

Singular form of key for resource.

resources_key = 'share_network_subnets'

Plural form of key for resource.

base_path = '/share-networks/%(share_network_id)s/subnets'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = False

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

share_network_id

Properties The share nerwork ID, part of the URI for share network subnets.

availability_zone

The name of the availability zone that the share network subnet belongs to.

cidr

The IP block from which to allocate the network, in CIDR notation.

created_at

Date and time the share network subnet was created at.

gateway

The gateway of a share network subnet.

ip_version

The IP version of the network.

mtu

The MTU of a share network subnet.

network_type

The network type. A valid value is VLAN, VXLAN, GRE, or flat

neutron_net_id

The name of the neutron network.

neutron_subnet_id

The ID of the neitron subnet.

segmentation_id

The segmentation ID.

share_network_name

The name of the share network that the share network subnet belongs to.

updated_at

Date and time the share network subnet was last updated at.

create(*session*, **args*, *resource_request_key='share-network-subnet'*, ***kwargs*)

Create a remote resource based on this instance.

Parameters

- **session** (Adapter) The session to use for making this request.
- **prepend_key** A boolean indicating whether the resource_key should be prepended in a resource creation request. Default to True.
- **base_path** (*str*) Base part of the URI for creating resources, if different from *base_path*.
- **resource_request_key** (*str*) Overrides the usage of self.resource_key when prepending a key to the request body. Ignored if *prepend_key* is false.
- **resource_response_key** (*str*) Overrides the usage of self.resource_key when processing response bodies. Ignored if *prepend_key* is false.
- **microversion** (*str*) API version to override the negotiated one.
- **params** (*dict*) Additional params to pass.

Returns

This Resource instance.

Raises

MethodNotSupported if Resource.allow_create is not set to True.

openstack.shared_file_system.v2.share_snapshot

The ShareSnapshot Class

The ShareSnapshot class inherits from *Resource*.

class openstack.shared_file_system.v2.share_snapshot.**ShareSnapshot**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of self._connection in Resource code should protect itself with a check for None.

resource_key = 'snapshot'

Singular form of key for resource.

resources_key = 'snapshots'

Plural form of key for resource.

base_path = '/snapshots'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_head = False

Allow head operation for this resource.

created_at

Properties The date and time stamp when the resource was created within the services database.

description

The user defined description of the resource.

display_name

The user defined name of the resource.

display_description

The user defined description of the resource

project_id

ID of the project that the snapshot belongs to.

share_id

The UUID of the source share that was used to create the snapshot.

share_proto

The file system protocol of a share snapshot

share_size

The snapshots source shares size, in GiBs.

size

The snapshot size, in GiBs.

status

The snapshot status

user_id

ID of the user that the snapshot was created by.

openstack.shared_file_system.v2.share_snapshot_instance

The ShareSnapshotInstance Class

The ShareSnapshotInstance class inherits from *Resource*.

class openstack.shared_file_system.v2.share_snapshot_instance.**ShareSnapshotInstance**(*_synchron* *con-* *nec-* *tion=Non* ***at-* *trs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'snapshot_instance'

Singular form of key for resource.

resources_key = 'snapshot_instances'

Plural form of key for resource.

base_path = '/snapshot-instances'

The base part of the URI for this resource.

allow_create = False

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = False

Allow update operation for this resource.

allow_delete = False

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_head = False

Allow head operation for this resource.

created_at

Properties The date and time stamp when the resource was created within the services database.

progress

The progress of the snapshot creation.

provider_location

Provider location of the snapshot on the backend.

share_id

The UUID of the share.

share_instance_id

The UUID of the share instance.

snapshot_id

The UUID of the snapshot.

status

The snapshot instance status.

updated_at

The date and time stamp when the resource was updated within the services database.

openstack.shared_file_system.v2.share_network

The ShareNetwork Class

The `ShareNetwork` class inherits from *Resource*.

class openstack.shared_file_system.v2.share_network.**ShareNetwork**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'share_network'

Singular form of key for resource.

resources_key = 'share_networks'

Plural form of key for resource.

base_path = '/share-networks'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_head = False

Allow head operation for this resource.

created_at

Properties The date and time stamp when the resource was created within the services database.

description

The user defined description of the resource.

project_id

The ID of the project that owns the resource.

share_network_subnets

A list of share network subnets that pertain to the related share network.

neutron_net_id

The UUID of a neutron network when setting up or updating a share network subnet with neutron.

neutron_subnet_id

The UUID of the neutron subnet when setting up or updating a share network subnet with neutron.

updated_at

The date and time stamp when the resource was last updated within the services database.

openstack.shared_file_system.v2.user_message

The UserMessage Class

The `UserMessage` class inherits from *Resource*.

class openstack.shared_file_system.v2.user_message.**UserMessage**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.

- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'message'

Singular form of key for resource.

resources_key = 'messages'

Plural form of key for resource.

base_path = '/messages'

The base part of the URI for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = False

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_head = False

Allow head operation for this resource.

action_id

Properties The action ID of the user message

created_at

Indicate when the user message was created

detail_id

The detail ID of the user message

expires_at

Indicate when the share message expires

message_level

The message level of the user message

project_id

The project ID of the user message

request_id

The request ID of the user message

resource_id

The resource ID of the user message

resource_type

The resource type of the user message

user_message

The message for the user message

openstack.shared_file_system.v2.share_group

The ShareGroup Class

The ShareGroup class inherits from *Resource*.

class openstack.shared_file_system.v2.share_group.**ShareGroup**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of self._connection in Resource code should protect itself with a check for None.

resource_key = 'share_group'

Singular form of key for resource.

resources_key = 'share_groups'

Plural form of key for resource.

base_path = '/share-groups'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_head = False

Allow head operation for this resource.

availability_zone

Properties The availability zone ID that the share group exists within.

consistent_snapshot_support

The consistency snapshot support.

created_at

The date and time stamp when the resource was created within the services database.

description

The user defined description of the resource.

project_id

The ID of the project that owns the resource.

share_group_snapshot_id

The share group snapshot ID.

share_group_type_id

The share group type ID.

share_network_id

The share network ID where the resource is exported to.

share_types

A list of share type IDs.

status

The share status

openstack.shared_file_system.v2.share_access_rule

The ShareAccessRule Class

The `ShareAccessRule` class inherits from *Resource*.

class openstack.shared_file_system.v2.share_access_rule.**ShareAccessRule**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'access'

Singular form of key for resource.

resources_key = 'access_list'

Plural form of key for resource.

base_path = '/share-access-rules'

The base part of the URI for this resource.

`allow_create = True`
: Allow create operation for this resource.

`allow_fetch = True`
: Allow get operation for this resource.

`allow_commit = False`
: Allow update operation for this resource.

`allow_delete = True`
: Allow delete operation for this resource.

`allow_list = True`
: Allow list operation for this resource.

`allow_head = False`
: Allow head operation for this resource.

`access_key`
: Properties The access credential of the entity granted share access.

`access_level`
: The access level to the share.

`access_list`
: The object of the access rule.

`access_to`
: The value that defines the access.

`access_type`
: The access rule type.

`created_at`
: The date and time stamp when the resource was created within the services database.

`metadata`
: One or more access rule metadata key and value pairs as a dictionary of strings.

`share_id`
: The UUID of the share to which you are granted or denied access.

`state`
: The state of the access rule.

`updated_at`
: The date and time stamp when the resource was last updated within the services database.

`lock_visibility`
: Whether the visibility of some sensitive fields is restricted or not

`lock_deletion`
: Whether the deletion of the access rule should be restricted or not

`lock_reason`
: Reason for placing the loc

create(*session*, **args*, ***kwargs*)

Create a remote resource based on this instance.

Parameters

- **session** (Adapter) The session to use for making this request.
- **prepend_key** A boolean indicating whether the resource_key should be prepended in a resource creation request. Default to True.
- **base_path** (*str*) Base part of the URI for creating resources, if different from *base_path*.
- **resource_request_key** (*str*) Overrides the usage of self.resource_key when prepending a key to the request body. Ignored if *prepend_key* is false.
- **resource_response_key** (*str*) Overrides the usage of self.resource_key when processing response bodies. Ignored if *prepend_key* is false.
- **microversion** (*str*) API version to override the negotiated one.
- **params** (*dict*) Additional params to pass.

Returns

This Resource instance.

Raises

MethodNotSupported if Resource.allow_create is not set to True.

delete(*session*, *error_message=None*, ***, *microversion=None*, *unrestrict=False*, ***kwargs*)

Delete the remote resource based on this instance.

Parameters

- **session** (Adapter) The session to use for making this request.
- **microversion** (*str*) API version to override the negotiated one.
- **kwargs** (*dict*) Parameters that will be passed to _prepare_request()

Returns

This Resource instance.

Raises

MethodNotSupported if Resource.allow_commit is not set to True.

Raises

NotFoundException if the resource was not found.

openstack.shared_file_system.v2.share_group_snapshot

The ShareGroupSnapshot Class

The ShareGroupSnapshot class inherits from *Resource*.

class openstack.shared_file_system.v2.share_group_snapshot.**ShareGroupSnapshot**(*_synchronized=F*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'share_group_snapshot'

Singular form of key for resource.

resources_key = 'share_group_snapshots'

Plural form of key for resource.

base_path = '/share-group-snapshots'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_head = False

Allow head operation for this resource.

project_id

Properties The ID of the project that owns the resource.

status

Filters by a share group snapshot status. A valid value is creating, error, available, deleting, error_deleting.

share_group_id

The UUID of the share group.

description

The user defined description of the resource.

created_at

The date and time stamp when the resource was created.

members

The share group snapshot members.

size

The snapshot size, in GiBs.

share_protocol

NFS, CIFS, GlusterFS, HDFS, CephFS or MAPRFS.

openstack.shared_file_system.v2.resource_locks

The Resource Locks Class

The ResourceLock class inherits from *Resource*.

class openstack.shared_file_system.v2.resource_locks.**ResourceLock**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of self._connection in Resource code should protect itself with a check for None.

resource_key = 'resource_lock'

Singular form of key for resource.

resources_key = 'resource_locks'

Plural form of key for resource.

base_path = '/resource-locks'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = True

Allow delete operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_head = False

Allow head operation for this resource.

created_at

Properties The date and time stamp when the resource was created within the services database.

updated_at

The date and time stamp when the resource was last modified within the services database.

user_id

The ID of the user that owns the lock

project_id

The ID of the project that owns the lock.

resource_type

The type of the resource that is locked, i.e.: share, access rule.

resource_id

The UUID of the resource that is locked.

resource_action

What action is currently locked, i.e.: deletion, visibility of fields.

lock_reason

The reason specified while the lock was being placed.

lock_context

The context that placed the lock (user, admin or service).

openstack.shared_file_system.v2.quota_class_set

The QuotaClassSet Class

The `QuotaClassSet` class inherits from *Resource* and can be used to query quota class

class openstack.shared_file_system.v2.quota_class_set.**QuotaClassSet**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

base_path = '/quota-class-sets'

The base part of the URI for this resource.

resource_key = 'quota_class_set'

Singular form of key for resource.

allow_create = False

Allow create operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_delete = False

Allow delete operation for this resource.

allow_list = False

Allow list operation for this resource.

allow_head = False

Allow head operation for this resource.

id

Properties A quota_class_set id.

share_groups

The maximum number of share groups.

share_group_snapshots

The maximum number of share group snapshots.

snapshots

The total maximum number of shares that are allowed in a project.

snapshot_gigabytes

The maximum number of snapshot gigabytes that are allowed in a project.

shares

The total maximum number of snapshot gigabytes that are allowed in a project.

share_networks

The maximum number of share-networks that are allowed in a project.

share_replicas

The maximum number of share replicas that is allowed.

gigabytes

The total maximum number of share gigabytes that are allowed in a project. You cannot request a share that exceeds the allowed gigabytes quota.

replica_gigabytes

The maximum number of replica gigabytes that are allowed in a project. You cannot create a share, share replica, manage a share or extend a share if it is going to exceed the allowed replica gigabytes quota.

per_share_gigabytes

The number of gigabytes per share allowed in a project.

backups

The total maximum number of share backups that are allowed in a project.

backup_gigabytes

The total maximum number of backup gigabytes that are allowed in a project.

Workflow Resources

openstack.workflow.v2.execution

The Execution Class

The `Execution` class inherits from *Resource*.

class openstack.workflow.v2.execution.**Execution**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'execution'

Singular form of key for resource.

resources_key = 'executions'

Plural form of key for resource.

base_path = '/executions'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_delete = True

Allow delete operation for this resource.

workflow_name

The name of the workflow

workflow_id

The ID of the workflow

description

A description of the workflow execution

task_execution_id

A reference to the parent task execution

status

Status can be one of: IDLE, RUNNING, SUCCESS, ERROR, or PAUSED

status_info

An optional information string about the status

params

An optional JSON structure containing workflow type specific parameters

output

The output of the workflow

created_at

The time at which the Execution was created

updated_at

The time at which the Execution was updated

create(*session*, *prepend_key=True*, *base_path=None*, ***kwargs*)

Create a remote resource based on this instance.

Parameters

- **session** (Adapter) The session to use for making this request.
- **prepend_key** A boolean indicating whether the resource_key should be prepended in a resource creation request. Default to True.
- **base_path** (*str*) Base part of the URI for creating resources, if different from *base_path*.
- **resource_request_key** (*str*) Overrides the usage of self.resource_key when prepending a key to the request body. Ignored if *prepend_key* is false.
- **resource_response_key** (*str*) Overrides the usage of self.resource_key when processing response bodies. Ignored if *prepend_key* is false.
- **microversion** (*str*) API version to override the negotiated one.
- **params** (*dict*) Additional params to pass.

Returns

This Resource instance.

Raises

MethodNotSupported if Resource.allow_create is not set to True.

openstack.workflow.v2.workflow

The Workflow Class

The Workflow class inherits from *Resource*.

class openstack.workflow.v2.workflow.**Workflow**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'workflow'

Singular form of key for resource.

resources_key = 'workflows'

Plural form of key for resource.

base_path = '/workflows'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_commit = True

Allow update operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_delete = True

Allow delete operation for this resource.

name

The name of this Workflow

input

The inputs for this Workflow

definition

A Workflow definition using the Mistral v2 DSL

scope

Can be either private or public

project_id

The ID of the associated project

created_at

The time at which the workflow was created

updated_at

The time at which the workflow was created

create(*session*, *prepend_key=True*, *base_path=None*, ***kwargs*)

Create a remote resource based on this instance.

Parameters

- **session** (Adapter) The session to use for making this request.
- **prepend_key** A boolean indicating whether the resource_key should be prepended in a resource creation request. Default to True.
- **base_path** (*str*) Base part of the URI for creating resources, if different from *base_path*.
- **resource_request_key** (*str*) Overrides the usage of self.resource_key when prepending a key to the request body. Ignored if *prepend_key* is false.
- **resource_response_key** (*str*) Overrides the usage of self.resource_key when processing response bodies. Ignored if *prepend_key* is false.
- **microversion** (*str*) API version to override the negotiated one.
- **params** (*dict*) Additional params to pass.

Returns

This Resource instance.

Raises

MethodNotSupported if Resource.allow_create is not set to True.

commit(*session*, *prepend_key=True*, *has_body=True*, *retry_on_conflict=None*, *base_path=None*, ***, *microversion=None*, ***kwargs*)

Commit the state of the instance to the remote resource.

Parameters

- **session** (Adapter) The session to use for making this request.
- **prepend_key** A boolean indicating whether the resource_key should be prepended in a resource update request. Default to True.
- **retry_on_conflict** (*bool*) Whether to enable retries on HTTP CONFLICT (409). Value of None leaves the *Adapter* defaults.
- **base_path** (*str*) Base part of the URI for modifying resources, if different from *base_path*.
- **microversion** (*str*) API version to override the negotiated one.
- **kwargs** (*dict*) Parameters that will be passed to _prepare_request()

Returns

This Resource instance.

Raises

MethodNotSupported if Resource.allow_commit is not set to True.

openstack.workflow.v2.cron_trigger

The CronTrigger Class

The `CronTrigger` class inherits from *Resource*.

class openstack.workflow.v2.cron_trigger.**CronTrigger**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = 'cron_trigger'

Singular form of key for resource.

resources_key = 'cron_triggers'

Plural form of key for resource.

base_path = '/cron_triggers'

The base part of the URI for this resource.

allow_create = True

Allow create operation for this resource.

allow_list = True

Allow list operation for this resource.

allow_fetch = True

Allow get operation for this resource.

allow_delete = True

Allow delete operation for this resource.

name

The name of this Cron Trigger

pattern

The pattern for this Cron Trigger

remaining_executions

Count of remaining exectuions

first_execution_time

Time of the first execution

next_execution_time

Time of the next execution

workflow_name

Workflow name

workflow_id

Workflow ID

workflow_input

The inputs for Workflow

workflow_params

Workflow params

project_id

The ID of the associated project

created_at

The time at which the cron trigger was created

updated_at

The time at which the cron trigger was created

create(*session*, *prepend_key=False*, **args*, ***kwargs*)

Create a remote resource based on this instance.

Parameters

- **session** (Adapter) The session to use for making this request.
- **prepend_key** A boolean indicating whether the resource_key should be prepended in a resource creation request. Default to True.
- **base_path** (*str*) Base part of the URI for creating resources, if different from *base_path*.
- **resource_request_key** (*str*) Overrides the usage of self.resource_key when prepending a key to the request body. Ignored if *prepend_key* is false.
- **resource_response_key** (*str*) Overrides the usage of self.resource_key when processing response bodies. Ignored if *prepend_key* is false.
- **microversion** (*str*) API version to override the negotiated one.
- **params** (*dict*) Additional params to pass.

Returns

This Resource instance.

Raises

MethodNotSupported if Resource.allow_create is not set to True.

Low-Level Classes

The following classes are not commonly used by application developers, but are used to construct applications to talk to OpenStack APIs. Typically these parts are managed through the *Connection Interface*, but their use can be customized.

Resource

Base resource class.

The *Resource* class is a base class that represent a remote resource. The attributes that comprise a request or response for this resource are specified as class members on the Resource subclass where their values are of a component type, including `Body`, `Header`, and `URI`.

For update management, *Resource* employs a series of `_ComponentManager` instances to look after the attributes of that particular component type. This is particularly useful for Body and Header types, so that only the values necessary are sent in requests to the server.

When making requests, each of the managers are looked at to gather the necessary URI, body, and header data to build a request to be sent via keystoneauths sessions. Responses from keystoneauth are then converted into this Resource class appropriate components and types and then returned to the caller.

Components

class openstack.resource.**Body**(*name*, *type=None*, *default=None*, *alias=None*, *aka=None*, *alternate_id=False*, *list_type=None*, *coerce_to_default=False*, *deprecated=False*, *deprecation_reason=None*)

class openstack.resource.**Header**(*name*, *type=None*, *default=None*, *alias=None*, *aka=None*, *alternate_id=False*, *list_type=None*, *coerce_to_default=False*, *deprecated=False*, *deprecation_reason=None*)

class openstack.resource.**URI**(*name*, *type=None*, *default=None*, *alias=None*, *aka=None*, *alternate_id=False*, *list_type=None*, *coerce_to_default=False*, *deprecated=False*, *deprecation_reason=None*)

The Resource class

class openstack.resource.**Resource**(*_synchronized=False*, *connection=None*, ***attrs*)

The base resource

Parameters

- **_synchronized** (*bool*) This is not intended to be used directly. See *new()* and *existing()*.
- **connection** (openstack.connection.Connection) Reference to the Connection being used. Defaults to None to allow Resource objects to be used without an active Connection, such as in unit tests. Use of `self._connection` in Resource code should protect itself with a check for None.

resource_key = None

Singular form of key for resource.

resources_key = None

Plural form of key for resource.

pagination_key = None

Key used for pagination links

id

The ID of this resource.

name

The name of this resource.

location

The OpenStack location of this resource.

base_path = ''

The base part of the URI for this resource.

allow_create = False

Allow create operation for this resource.

allow_fetch = False

Allow get operation for this resource.

allow_commit = False

Allow update operation for this resource.

allow_delete = False

Allow delete operation for this resource.

allow_list = False

Allow list operation for this resource.

allow_head = False

Allow head operation for this resource.

allow_patch = False

Allow patch operation for this resource.

allow_empty_commit = False

Commits happen without header or body being dirty.

commit_method = 'PUT'

Method for committing a resource (PUT, PATCH, POST)

create_method = 'POST'

Method for creating a resource (POST, PUT)

commit_jsonpatch = False

Whether commit uses JSON patch format.

requires_id = True

Do calls for this resource require an id

create_requires_id = None

Whether create requires an ID (determined from method if None).

create_exclude_id_from_body = False

Whether create should exclude ID in the body of the request.

has_body = True

Do responses for this resource have bodies

create_returns_body = None

Does create returns a body (if False requires ID), defaults to has_body

microversion = None

API microversion (string or None) this Resource was loaded with

keys() → a set-like object providing a view on D's keys

items() → a set-like object providing a view on D's items

classmethod new(***kwargs*)

Create a new instance of this resource.

When creating the instance set the `_synchronized` parameter of *Resource* to `False` to indicate that the resource does not yet exist on the server side. This marks all attributes passed in `**kwargs` as dirty on the resource, and thusly tracked as necessary in subsequent calls such as `update()`.

Parameters

kwargs (*dict*) Each of the named arguments will be set as attributes on the resulting Resource object.

classmethod existing(*connection=None*, ***kwargs*)

Create an instance of an existing remote resource.

When creating the instance set the `_synchronized` parameter of *Resource* to `True` to indicate that it represents the state of an existing server-side resource. As such, all attributes passed in `**kwargs` are considered clean, such that an immediate `update()` call would not generate a body of attributes to be modified on the server.

Parameters

kwargs (*dict*) Each of the named arguments will be set as attributes on the resulting Resource object.

to_dict(*body=True*, *headers=True*, *computed=True*, *ignore_none=False*, *original_names=False*, *_to_munch=False*)

Return a dictionary of this resources contents

Parameters

- **body** (*bool*) Include the `Body` attributes in the returned dictionary.
- **headers** (*bool*) Include the `Header` attributes in the returned dictionary.
- **computed** (*bool*) Include the `Computed` attributes in the returned dictionary.
- **ignore_none** (*bool*) When True, exclude key/value pairs where the value is None. This will exclude attributes that the server hasnt returned.
- **original_names** (*bool*) When True, use attribute names as they were received from the server.
- **_to_munch** (*bool*) For internal use only. Converts to *utils.Munch* instead of dict.

Returns

A dictionary of key/value pairs where keys are named as they exist as attributes of this class.

toDict(*body=True*, *headers=True*, *computed=True*, *ignore_none=False*, *original_names=False*, *_to_munch=False*)

Return a dictionary of this resources contents

Parameters

- **body** (*bool*) Include the Body attributes in the returned dictionary.
- **headers** (*bool*) Include the Header attributes in the returned dictionary.
- **computed** (*bool*) Include the Computed attributes in the returned dictionary.
- **ignore_none** (*bool*) When True, exclude key/value pairs where the value is None. This will exclude attributes that the server hasnt returned.
- **original_names** (*bool*) When True, use attribute names as they were received from the server.
- **_to_munch** (*bool*) For internal use only. Converts to *utils.Munch* instead of dict.

Returns

A dictionary of key/value pairs where keys are named as they exist as attributes of this class.

copy(*body=True*, *headers=True*, *computed=True*, *ignore_none=False*, *original_names=False*, *_to_munch=False*)

Return a dictionary of this resources contents

Parameters

- **body** (*bool*) Include the Body attributes in the returned dictionary.
- **headers** (*bool*) Include the Header attributes in the returned dictionary.
- **computed** (*bool*) Include the Computed attributes in the returned dictionary.
- **ignore_none** (*bool*) When True, exclude key/value pairs where the value is None. This will exclude attributes that the server hasnt returned.
- **original_names** (*bool*) When True, use attribute names as they were received from the server.
- **_to_munch** (*bool*) For internal use only. Converts to *utils.Munch* instead of dict.

Returns

A dictionary of key/value pairs where keys are named as they exist as attributes of this class.

create(*session*, *prepend_key=True*, *base_path=None*, ***, *resource_request_key=None*, *resource_response_key=None*, *microversion=None*, ***params*)

Create a remote resource based on this instance.

Parameters

- **session** (Adapter) The session to use for making this request.

- **prepend_key** A boolean indicating whether the resource_key should be prepended in a resource creation request. Default to True.
- **base_path** (*str*) Base part of the URI for creating resources, if different from *base_path*.
- **resource_request_key** (*str*) Overrides the usage of self.resource_key when prepending a key to the request body. Ignored if *prepend_key* is false.
- **resource_response_key** (*str*) Overrides the usage of self.resource_key when processing response bodies. Ignored if *prepend_key* is false.
- **microversion** (*str*) API version to override the negotiated one.
- **params** (*dict*) Additional params to pass.

Returns
This *Resource* instance.

Raises
MethodNotSupported if *Resource.allow_create* is not set to True.

classmethod bulk_create(*session*, *data*, *prepend_key=True*, *base_path=None*, ***, *microversion=None*, ***params*)

Create multiple remote resources based on this class and data.

Parameters

- **session** (Adapter) The session to use for making this request.
- **data** list of dicts, which represent resources to create.
- **prepend_key** A boolean indicating whether the resource_key should be prepended in a resource creation request. Default to True.
- **base_path** (*str*) Base part of the URI for creating resources, if different from *base_path*.
- **microversion** (*str*) API version to override the negotiated one.
- **params** (*dict*) Additional params to pass.

Returns
A generator of *Resource* objects.

Raises
MethodNotSupported if *Resource.allow_create* is not set to True.

fetch(*session*, *requires_id=True*, *base_path=None*, *error_message=None*, *skip_cache=False*, ***, *resource_response_key=None*, *microversion=None*, ***params*)

Get a remote resource based on this instance.

Parameters

- **session** (Adapter) The session to use for making this request.
- **requires_id** (*boolean*) A boolean indicating whether resource ID should be part of the requested URI.
- **base_path** (*str*) Base part of the URI for fetching resources, if different from *base_path*.

- **error_message** (*str*) An Error message to be returned if requested object does not exist.
- **skip_cache** (*bool*) A boolean indicating whether optional API cache should be skipped for this invocation.
- **resource_response_key** (*str*) Overrides the usage of self.resource_key when processing the response body.
- **microversion** (*str*) API version to override the negotiated one.
- **params** (*dict*) Additional parameters that can be consumed.

Returns
This *Resource* instance.

Raises
MethodNotSupported if *Resource.allow_fetch* is not set to `True`.

Raises
NotFoundException if the resource was not found.

head(*session*, *base_path=None*, ***, *microversion=None*)

Get headers from a remote resource based on this instance.

Parameters

- **session** (`Adapter`) The session to use for making this request.
- **base_path** (*str*) Base part of the URI for fetching resources, if different from *base_path*.
- **microversion** (*str*) API version to override the negotiated one.

Returns
This *Resource* instance.

Raises
MethodNotSupported if *Resource.allow_head* is not set to `True`.

Raises
NotFoundException if the resource was not found.

property requires_commit

Whether the next commit() call will do anything.

commit(*session*, *prepend_key=True*, *has_body=True*, *retry_on_conflict=None*, *base_path=None*, ***, *microversion=None*, ***kwargs*)

Commit the state of the instance to the remote resource.

Parameters

- **session** (`Adapter`) The session to use for making this request.
- **prepend_key** A boolean indicating whether the resource_key should be prepended in a resource update request. Default to True.
- **retry_on_conflict** (*bool*) Whether to enable retries on HTTP CONFLICT (409). Value of `None` leaves the *Adapter* defaults.
- **base_path** (*str*) Base part of the URI for modifying resources, if different from *base_path*.

- **microversion** (*str*) API version to override the negotiated one.
- **kwargs** (*dict*) Parameters that will be passed to _prepare_request()

Returns
This *Resource* instance.

Raises
MethodNotSupported if *Resource.allow_commit* is not set to `True`.

patch(*session*, *patch=None*, *prepend_key=True*, *has_body=True*, *retry_on_conflict=None*, *base_path=None*, ***, *microversion=None*)

Patch the remote resource.

Allows modifying the resource by providing a list of JSON patches to apply to it. The patches can use both the original (server-side) and SDK field names.

Parameters

- **session** (`Adapter`) The session to use for making this request.
- **patch** Additional JSON patch as a list or one patch item. If provided, it is applied on top of any changes to the current resource.
- **prepend_key** A boolean indicating whether the resource_key should be prepended in a resource update request. Default to True.
- **retry_on_conflict** (*bool*) Whether to enable retries on HTTP CONFLICT (409). Value of `None` leaves the *Adapter* defaults.
- **base_path** (*str*) Base part of the URI for modifying resources, if different from *base_path*.
- **microversion** (*str*) API version to override the negotiated one.

Returns
This *Resource* instance.

Raises
MethodNotSupported if *Resource.allow_patch* is not set to `True`.

delete(*session*, *error_message=None*, ***, *microversion=None*, ***kwargs*)

Delete the remote resource based on this instance.

Parameters

- **session** (`Adapter`) The session to use for making this request.
- **microversion** (*str*) API version to override the negotiated one.
- **kwargs** (*dict*) Parameters that will be passed to _prepare_request()

Returns
This *Resource* instance.

Raises
MethodNotSupported if *Resource.allow_commit* is not set to `True`.

Raises
NotFoundException if the resource was not found.

classmethod list(*session*, *paginated=True*, *base_path=None*, *allow_unknown_params=False*, ***, *microversion=None*, *headers=None*, ***params*)

This method is a generator which yields resource objects.

This resource object list generator handles pagination and takes query params for response filtering.

Parameters

- **session** (Adapter) The session to use for making this request.
- **paginated** (*bool*) True if a GET to this resource returns a paginated series of responses, or False if a GET returns only one page of data. **When paginated is False only one page of data will be returned regardless of the APIs support of pagination.**
- **base_path** (*str*) Base part of the URI for listing resources, if different from *base_path*.
- **allow_unknown_params** (*bool*) True to accept, but discard unknown query parameters. This allows getting list of filters and passing everything known to the server. False will result in validation exception when unknown query parameters are passed.
- **microversion** (*str*) API version to override the negotiated one.
- **headers** (*dict*) Additional headers to inject into the HTTP request.
- **params** (*dict*) These keyword arguments are passed through the _transpose() method to find if any of them match expected query parameters to be sent in the *params* argument to get(). They are additionally checked against the *base_path* format string to see if any path fragments need to be filled in by the contents of this argument. Parameters supported as filters by the server side are passed in the API call, remaining parameters are applied as filters to the retrieved results.

Returns

A generator of *Resource* objects.

Raises

MethodNotSupported if *Resource.allow_list* is not set to True.

Raises

InvalidResourceQuery if query contains invalid params.

classmethod find(*session*, *name_or_id*, *ignore_missing=True*, *list_base_path=None*, ***, *microversion=None*, *all_projects=None*, ***params*)

Find a resource by its name or id.

Parameters

- **session** (Adapter) The session to use for making this request.
- **name_or_id** This resources identifier, if needed by the request. The default is None.

- **ignore_missing** (*bool*) When set to `False` *NotFoundException* will be raised when the resource does not exist. When set to `True`, None will be returned when attempting to find a nonexistent resource.
- **list_base_path** (*str*) base_path to be used when need listing resources.
- **microversion** (*str*) API version to override the negotiated one.
- **params** (*dict*) Any additional parameters to be passed into underlying methods, such as to *existing()* in order to pass on URI parameters.

Returns

The *Resource* object matching the given name or id or None if nothing matches.

Raises

openstack.exceptions.DuplicateResource if more than one resource is found for this request.

Raises

openstack.exceptions.NotFoundException if nothing is found and ignore_missing is `False`.

ServiceDescription

ServiceDescription object

`class` openstack.service_description.**ServiceDescription**(*service_type*, *supported_versions=None*, *aliases=None*)

Class describing how to interact with a REST service.

Each service in an OpenStack cloud needs to be found by looking for it in the catalog. Once the endpoint is found, REST calls can be made, but a Proxy class and some Resource objects are needed to provide an object interface.

Instances of ServiceDescription can be passed to *openstack.connection.Connection.add_service*, or a list can be passed to the *openstack.connection.Connection* constructor in the `extra_services` argument.

All three parameters can be provided at instantation time, or a service-specific subclass can be used that sets the attributes directly.

Parameters

- **service_type** (*string*) service_type to look for in the keystone catalog
- **aliases** (*list*) Optional list of aliases, if there is more than one name that might be used to register the service in the catalog.

service_type

main service_type to use to find this service in the catalog

supported_versions = {}

Dictionary of supported versions and proxy classes for that version

aliases = []

list of aliases this service might be registered as

Utilities

Errors and warnings

The SDK attempts to provide detailed errors and warnings for things like failed requests, deprecated APIs, and invalid configurations. Application developers are responsible for handling these errors and can opt into warnings to ensure their applications stay up-to-date.

Exceptions

openstacksdk provides a number of exceptions for commonly encountered issues, such as missing API endpoints, various HTTP error codes, timeouts and so forth. It is the responsibility of the calling application to handle these exceptions appropriately.

Available exceptions

Exception definitions.

exception openstack.exceptions.**SDKException**(*message=None*, *extra_data=None*)

The base exception class for all exceptions this library raises.

exception openstack.exceptions.**EndpointNotFound**(*message=None*)

A mismatch occurred between what the client and server expect.

exception openstack.exceptions.**InvalidResponse**(*message=None*)

The response from the server is not valid for this request.

exception openstack.exceptions.**InvalidRequest**(*message=None*)

The request to the server is not valid.

exception openstack.exceptions.**HttpException**(*message='Error'*, *response=None*, *http_status=None*, *details=None*, *request_id=None*)

The base exception for all HTTP error responses.

Initialize RequestException with *request* and *response* objects.

exception openstack.exceptions.**BadRequestException**(*message='Error'*, *response=None*, *http_status=None*, *details=None*, *request_id=None*)

HTTP 400 Bad Request.

Initialize RequestException with *request* and *response* objects.

exception openstack.exceptions.**NotFoundException**(*message='Error'*, *response=None*, *http_status=None*, *details=None*, *request_id=None*)

HTTP 404 Not Found.

Initialize RequestException with *request* and *response* objects.

exception openstack.exceptions.**ForbiddenException**(*message='Error'*, *response=None*, *http_status=None*, *details=None*, *request_id=None*)

HTTP 403 Forbidden Request.

Initialize RequestException with *request* and *response* objects.

exception openstack.exceptions.**ConflictException**(*message='Error'*, *response=None*, *http_status=None*, *details=None*, *request_id=None*)

HTTP 409 Conflict.

Initialize RequestException with *request* and *response* objects.

exception openstack.exceptions.**PreconditionFailedException**(*message='Error'*, *response=None*, *http_status=None*, *details=None*, *request_id=None*)

HTTP 412 Precondition Failed.

Initialize RequestException with *request* and *response* objects.

exception openstack.exceptions.**MethodNotSupported**(*resource*, *method*)

The resource does not support this operation type.

exception openstack.exceptions.**DuplicateResource**(*message=None*, *extra_data=None*)

More than one resource exists with that name.

exception openstack.exceptions.**ResourceTimeout**(*message=None*, *extra_data=None*)

Timeout waiting for resource.

exception openstack.exceptions.**ResourceFailure**(*message=None*, *extra_data=None*)

General resource failure.

exception openstack.exceptions.**InvalidResourceQuery**(*message=None*, *extra_data=None*)

Invalid query params for resource.

openstack.exceptions.**raise_from_response**(*response*, *error_message=None*)

Raise an instance of an HTTPException based on keystoneauth response.

exception openstack.exceptions.**ConfigException**(*message=None*, *extra_data=None*)

Something went wrong with parsing your OpenStack Config.

exception openstack.exceptions.**NotSupported**(*message=None*, *extra_data=None*)

Request cannot be performed by any supported API version.

exception openstack.exceptions.**ValidationException**(*message=None*, *extra_data=None*)

Validation failed for resource.

exception openstack.exceptions.**ServiceDisabledException**(*message=None*, *extra_data=None*)

This service is disabled for reasons.

exception openstack.exceptions.**ServiceDiscoveryException**(*message=None*, *extra_data=None*)

The service cannot be discovered.

openstack.exceptions.**OpenStackCloudException**

alias of *SDKException*

openstack.exceptions.**ResourceNotFound**

alias of *NotFoundException*

Warnings

openstacksdk uses the warnings infrastructure to warn users about deprecated resources and resource fields, as well as deprecated behavior in openstacksdk itself. These warnings are derived from `Warning` or `DeprecationWarning`. In Python, warnings are emitted by default while deprecation warnings are silenced by default and must be turned on using the `-Wa` Python command line option or the `PYTHONWARNINGS` environment variable. If you are writing an application that uses openstacksdk, you may wish to enable some of these warnings during test runs to ensure you migrate away from deprecated behavior.

Available warnings

exception openstack.warnings.**OpenStackDeprecationWarning**

Base class for warnings about deprecated features in openstacksdk.

exception openstack.warnings.**RemovedResourceWarning**

Indicates that a resource has been removed in newer API versions and should not be used.

exception openstack.warnings.**RemovedFieldWarning**

Indicates that a field has been removed in newer API versions and should not be used.

exception openstack.warnings.**LegacyAPIWarning**

Indicates an API that is in legacy status, a long term deprecation.

exception openstack.warnings.**RemovedInSDK50Warning**

Indicates an argument that is deprecated for removal in SDK 5.0.

exception openstack.warnings.**RemovedInSDK60Warning**

Indicates an argument that is deprecated for removal in SDK 6.0.

exception openstack.warnings.**OpenStackWarning**

Base class for general warnings in openstacksdk.

exception openstack.warnings.**ConfigurationWarning**

Indicates an issue with configuration.

exception openstack.warnings.**UnsupportedServiceVersion**

Indicates a major version that SDK doesnt understand.

2.1.4 Presentations

Multi-Cloud Demo

This document contains a presentation in presentty format. If you want to walk through it like a presentation, install *presentty* and run:

```
presentty doc/source/user/multi-cloud-demo.rst
```

The content is hopefully helpful even if its not being narrated, so its being included in the openstacksdk docs.

Who am I?

Monty Taylor

- OpenStack Infra Core
- irc: mordred
- twitter: @e_monty

What are we going to talk about?

OpenStackSDK

- a task and end-user oriented Python library
- abstracts deployment differences
- designed for multi-cloud
- simple to use
- massive scale
  - optional advanced features to handle 20k servers a day
- Initial logic/design extracted from nodepool
- Librified to re-use in Ansible

OpenStackSDK is Free Software

- https://opendev.org/openstack/openstacksdk
- openstack-discuss@lists.openstack.org
- #openstack-sdks on oftc

This talk is Free Software, too

- Written for presentty (https://pypi.org/project/presentty)
- doc/source/user/multi-cloud-demo.rst
- examples in examples/cloud
- Paths subject to change - this is the first presentation in tree!

Complete Example

```
from openstack import cloud as openstack

# Initialize and turn on debug logging
openstack.enable_logging(debug=True)

for cloud_name, region_name in [
    ('my-vexxhost', 'ca-ymq-1'),
    ('my-citycloud', 'Buf1'),
    ('my-internap', 'ams01'),
```

(continues on next page)

(continued from previous page)

```
]:
    # Initialize cloud
    cloud = openstack.connect(cloud=cloud_name, region_name=region_name)

    # Upload an image to the cloud
    image = cloud.create_image(
        'devuan-jessie',
        filename='devuan-jessie.qcow2',
        wait=True,
    )

    # Find a flavor with at least 512M of RAM
    flavor = cloud.get_flavor_by_ram(512)

    # Boot a server, wait for it to boot, and then do whatever is needed
    # to get a public ip for it.
    cloud.create_server(
        'my-server',
        image=image,
        flavor=flavor,
        wait=True,
        auto_ip=True,
    )
```

Lets Take a Few Steps Back

Multi-cloud is easy, but you need to know a few things.

- Terminology
- Config
- OpenStackSDK API

Cloud Terminology

Lets define a few terms, so that we can use them with ease:

- *cloud* - logically related collection of services
- *region* - completely independent subset of a given cloud
- *patron* - human who has an account
- *user* - account on a cloud
- *project* - logical collection of cloud resources
- *domain* - collection of users and projects

Cloud Terminology Relationships

- A *cloud* has one or more *regions*
- A *patron* has one or more *users*
- A *patron* has one or more *projects*
- A *cloud* has one or more *domains*
- In a *cloud* with one *domain* it is named default
- Each *patron* may have their own *domain*
- Each *user* is in one *domain*
- Each *project* is in one *domain*
- A *user* has one or more *roles* on one or more *projects*

HTTP Sessions

- HTTP interactions are authenticated via keystone
- Authenticating returns a *token*
- An authenticated HTTP Session is shared across a *region*

Cloud Regions

A *cloud region* is the basic unit of REST interaction.

- A *cloud* has a *service catalog*
- The *service catalog* is returned in the *token*
- The *service catalog* lists *endpoint* for each *service* in each *region*
- A *region* is completely autonomous

Users, Projects and Domains

In clouds with multiple domains, project and user names are only unique within a region.

- Names require *domain* information for uniqueness. IDs do not.
- Providing *domain* information when not needed is fine.
- *project_name* requires *project_domain_name* or *project_domain_id*
- *project_id* does not
- *username* requires *user_domain_name* or *user_domain_id*
- *user_id* does not

Confused Yet?

Dont worry - you dont have to deal with most of that.

Auth per cloud, select per region

In general, the thing you need to know is:

- Configure authentication per *cloud*
- Select config to use by *cloud* and *region*

clouds.yaml

Information about the clouds you want to connect to is stored in a file called *clouds.yaml*.

clouds.yaml can be in your homedir: *~/.config/openstack/clouds.yaml* or system-wide: */etc/openstack/clouds.yaml*.

Information in your homedir, if it exists, takes precedence.

Full docs on *clouds.yaml* are at https://docs.openstack.org/os-client-config/latest/

What about Mac and Windows?

USER_CONFIG_DIR is different on Linux, OSX and Windows.

- Linux: *~/.config/openstack*
- OSX: *~/Library/Application Support/openstack*
- Windows: *C:\Users\USERNAME\AppData\Local\OpenStack\openstack*

SITE_CONFIG_DIR is different on Linux, OSX and Windows.

- Linux: */etc/openstack*
- OSX: */Library/Application Support/openstack*
- Windows: *C:\ProgramData\OpenStack\openstack*

Config Terminology

For multi-cloud, think of two types:

- *profile* - Facts about the *cloud* that are true for everyone
- *cloud* - Information specific to a given *user*

Apologies for the use of *cloud* twice.

Environment Variables and Simple Usage

- Environment variables starting with *OS_* go into a cloud called *envvars*
- If you only have one cloud, you dont have to specify it
- *OS_CLOUD* and *OS_REGION_NAME* are default values for *cloud* and *region_name*

TOO MUCH TALKING - NOT ENOUGH CODE

basic clouds.yaml for the example code

Simple example of a clouds.yaml

- Config for a named *cloud* my-citycloud
- Reference a well-known named profile: *citycloud*
- *os-client-config* has a built-in list of profiles at https://docs.openstack.org/openstacksdk/latest/user/config/vendor-support.html
- Vendor profiles contain various advanced config
- *cloud* name can match *profile* name (using different names for clarity)

```
clouds:
  my-citycloud:
    profile: citycloud
    auth:
      username: mordred
      project_id: 65222a4d09ea4c68934fa1028c77f394
      user_domain_id: d0919bd5e8d74e49adf0e145807ffc38
      project_domain_id: d0919bd5e8d74e49adf0e145807ffc38
```

Wheres the password?

secure.yaml

- Optional additional file just like *clouds.yaml*
- Values overlaid on *clouds.yaml*
- Useful if you want to protect secrets more stringently

Example secure.yaml

- No, my password isnt XXXXXXXX
- *cloud* name should match *clouds.yaml*
- Optional - I actually keep mine in my *clouds.yaml*

```
clouds:
  my-citycloud:
    auth:
      password: XXXXXXXX
```

more clouds.yaml

More information can be provided.

- Use v3 of the *identity* API - even if others are present
- Use *https://image-ca-ymq-1.vexxhost.net/v2* for *image* API instead of whats in the catalog

```
my-vexxhost:
  identity_api_version: 3
  image_endpoint_override: https://image-ca-ymq-1.vexxhost.net/v2
  profile: vexxhost
  auth:
```

(continues on next page)

(continued from previous page)

```
    user_domain_id: default
    project_domain_id: default
    project_name: d8af8a8f-a573-48e6-898a-af333b970a2d
    username: 0b8c435b-cc4d-4e05-8a47-a2ada0539af1
```

Much more complex clouds.yaml example

- Not using a profile - all settings included
- In the *ams01 region* there are two networks with undiscoverable qualities
- Each one are labeled here so choices can be made
- Any of the settings can be specific to a *region* if needed
- *region* settings override *cloud* settings
- *cloud* does not support *floating-ips*

```
my-internap:
  auth:
    auth_url: https://identity.api.cloud.inap.com
    username: api-55f9a00fb2619
    project_name: inap-17037
  identity_api_version: 3
  floating_ip_source: None
  regions:
  - name: ams01
    values:
      networks:
      - name: inap-17037-WAN1654
        routes_externally: true
        default_interface: true
      - name: inap-17037-LAN3631
        routes_externally: false
```

Complete Example Again

```
from openstack import cloud as openstack

# Initialize and turn on debug logging
openstack.enable_logging(debug=True)

for cloud_name, region_name in [
        ('my-vexxhost', 'ca-ymq-1'),
        ('my-citycloud', 'Buf1'),
        ('my-internap', 'ams01')]:
    # Initialize cloud
    cloud = openstack.connect(cloud=cloud_name, region_name=region_name)

    # Upload an image to the cloud
```

(continues on next page)

(continued from previous page)

```
    image = cloud.create_image(
        'devuan-jessie', filename='devuan-jessie.qcow2', wait=True)

    # Find a flavor with at least 512M of RAM
    flavor = cloud.get_flavor_by_ram(512)

    # Boot a server, wait for it to boot, and then do whatever is needed
    # to get a public ip for it.
    cloud.create_server(
        'my-server', image=image, flavor=flavor, wait=True, auto_ip=True)
```

Step By Step

Import the library

```
from openstack import cloud as openstack
```

Logging

- *openstacksdk* uses standard python logging
- `openstack.enable_logging` does easy defaults
- Squelches some meaningless warnings
  - *debug*
    - Logs openstacksdk loggers at debug level
  - *http_debug* Implies *debug*, turns on HTTP tracing

```
# Initialize and turn on debug logging
openstack.enable_logging(debug=True)
```

Example with Debug Logging

- examples/cloud/debug-logging.py

```
from openstack import cloud as openstack
openstack.enable_logging(debug=True)

cloud = openstack.connect(cloud='my-vexxhost', region_name='ca-ymq-1')
cloud.get_image('Ubuntu 16.04.1 LTS [2017-03-03]')
```

Example with HTTP Debug Logging

- examples/cloud/http-debug-logging.py

```
from openstack import cloud as openstack
openstack.enable_logging(http_debug=True)
```

(continues on next page)

(continued from previous page)

```
cloud = openstack.connect(
    cloud='my-vexxhost', region_name='ca-ymq-1')
cloud.get_image('Ubuntu 16.04.1 LTS [2017-03-03]')
```

Cloud Regions

- *cloud* constructor needs *cloud* and *region_name*
- *openstack.connect* is a helper factory function

```
for cloud_name, region_name in [
    ('my-vexxhost', 'ca-ymq-1'),
    ('my-citycloud', 'Buf1'),
    ('my-internap', 'ams01')
]:
    # Initialize cloud
    cloud = openstack.connect(cloud=cloud_name, region_name=region_name)
```

Upload an Image

- Picks the correct upload mechanism
- **SUGGESTION** Always upload your own base images

```
# Upload an image to the cloud
image = cloud.create_image(
    'devuan-jessie',
    filename='devuan-jessie.qcow2',
    wait=True,
)
```

Always Upload an Image

Ok. You dont have to. But, for multi-cloud

- Images with same content are named different on different clouds
- Images with same name on different clouds can have different content
- Upload your own to all clouds, both problems go away
- Download from OS vendor or build with *diskimage-builder*

Find a flavor

- Flavors are all named differently on clouds
- Flavors can be found via RAM
- *get_flavor_by_ram* finds the smallest matching flavor

```
# Find a flavor with at least 512M of RAM
flavor = cloud.get_flavor_by_ram(512)
```

Create a server

- my-vexxhost
  - Boot server
  - Wait for *status==ACTIVE*
- my-internap
  - Boot server on network *inap-17037-WAN1654*
  - Wait for *status==ACTIVE*
- my-citycloud
  - Boot server
  - Wait for *status==ACTIVE*
  - Find the *port* for the *fixed_ip* for *server*
  - Create *floating-ip* on that *port*
  - Wait for *floating-ip* to attach

```
# Boot a server, wait for it to boot, and then do whatever is needed
# to get a public ip for it.
cloud.create_server(
    'my-server', image=image, flavor=flavor, wait=True, auto_ip=True)
```

Wow. We didnt even deploy Wordpress!

Image and Flavor by Name or ID

- Pass string to image/flavor
- Image/Flavor will be found by name or ID
- Common pattern
- examples/cloud/create-server-name-or-id.py

```
from openstack import cloud as openstack

# Initialize and turn on debug logging
openstack.enable_logging(debug=True)

for cloud_name, region_name, image, flavor in [
        ('my-vexxhost', 'ca-ymq-1',
         'Ubuntu 16.04.1 LTS [2017-03-03]', 'v1-standard-4'),
        ('my-citycloud', 'Buf1',
         'Ubuntu 16.04 Xenial Xerus', '4C-4GB-100GB'),
        ('my-internap', 'ams01',
         'Ubuntu 16.04 LTS (Xenial Xerus)', 'A1.4')]:
    # Initialize cloud
    cloud = openstack.connect(cloud=cloud_name, region_name=region_name)
```

(continues on next page)

(continued from previous page)

```
    # Boot a server, wait for it to boot, and then do whatever is needed
    # to get a public ip for it.
    server = cloud.create_server(
        'my-server', image=image, flavor=flavor, wait=True, auto_ip=True)
    print(server.name)
    print(server['name'])
    cloud.pprint(server)
    # Delete it - this is a demo
    cloud.delete_server(server, wait=True, delete_ips=True)
```

Delete Servers

- *delete_ips* Delete any *floating_ips* the server may have

```
cloud.delete_server('my-server', wait=True, delete_ips=True)
```

Image and Flavor by Dict

- Pass dict to image/flavor
- If you know if the value is Name or ID
- Common pattern
- examples/cloud/create-server-dict.py

```
from openstack import cloud as openstack

# Initialize and turn on debug logging
openstack.enable_logging(debug=True)

for cloud_name, region_name, image, flavor_id in [
        ('my-vexxhost', 'ca-ymq-1', 'Ubuntu 16.04.1 LTS [2017-03-03]',
         '5cf64088-893b-46b5-9bb1-ee020277635d'),
        ('my-citycloud', 'Buf1', 'Ubuntu 16.04 Xenial Xerus',
         '0dab10b5-42a2-438e-be7b-505741a7ffcc'),
        ('my-internap', 'ams01', 'Ubuntu 16.04 LTS (Xenial Xerus)',
         'A1.4')]:
    # Initialize cloud
    cloud = openstack.connect(cloud=cloud_name, region_name=region_name)

    # Boot a server, wait for it to boot, and then do whatever is needed
    # to get a public ip for it.
    server = cloud.create_server(
        'my-server', image=image, flavor=dict(id=flavor_id),
        wait=True, auto_ip=True)
    # Delete it - this is a demo
    cloud.delete_server(server, wait=True, delete_ips=True)
```

Munch Objects

- Behave like a dict and an object
- examples/cloud/munch-dict-object.py

```
from openstack import cloud as openstack
openstack.enable_logging(debug=True)

cloud = openstack.connect(cloud='zetta', region_name='no-osl1')
image = cloud.get_image('Ubuntu 14.04 (AMD64) [Local Storage]')
print(image.name)
print(image['name'])
```

API Organized by Logical Resource

- list_servers
- search_servers
- get_server
- create_server
- delete_server
- update_server

For other things, its still {verb}_{noun}

- attach_volume
- wait_for_server
- add_auto_ip

Cleanup Script

- Sometimes my examples had bugs
- examples/cloud/cleanup-servers.py

```
from openstack import cloud as openstack

# Initialize and turn on debug logging
openstack.enable_logging(debug=True)

for cloud_name, region_name in [
        ('my-vexxhost', 'ca-ymq-1'),
        ('my-citycloud', 'Buf1'),
        ('my-internap', 'ams01')]:
    # Initialize cloud
    cloud = openstack.connect(cloud=cloud_name, region_name=region_name)
    for server in cloud.search_servers('my-server'):
        cloud.delete_server(server, wait=True, delete_ips=True)
```

Normalization

- examples/cloud/normalization.py

```
from openstack import cloud as openstack
openstack.enable_logging()

cloud = openstack.connect(cloud='fuga', region_name='cystack')
image = cloud.get_image(
    'Ubuntu 16.04 LTS - Xenial Xerus - 64-bit - Fuga Cloud Based Image')
cloud.pprint(image)
```

Strict Normalized Results

- Return only the declared model
- examples/cloud/strict-mode.py

```
from openstack import cloud as openstack
openstack.enable_logging()

cloud = openstack.connect(
    cloud='fuga', region_name='cystack', strict=True)
image = cloud.get_image(
    'Ubuntu 16.04 LTS - Xenial Xerus - 64-bit - Fuga Cloud Based Image')
cloud.pprint(image)
```

How Did I Find the Image Name for the Last Example?

- I often make stupid little utility scripts
- examples/cloud/find-an-image.py

```
from openstack import cloud as openstack
openstack.enable_logging()

cloud = openstack.connect(cloud='fuga', region_name='cystack')
cloud.pprint([
    image for image in cloud.list_images()
    if 'ubuntu' in image.name.lower()])
```

Added / Modified Information

- Servers need more extra help
- Fetch addresses dict from neutron
- Figure out which IPs are good
- *detailed* - defaults to True, add everything
- *bare* - no extra calls - dont even fix broken things
- *bare* is still normalized

- examples/cloud/server-information.py

```
from openstack import cloud as openstack
openstack.enable_logging(debug=True)

cloud = openstack.connect(cloud='my-citycloud', region_name='Buf1')
try:
    server = cloud.create_server(
        'my-server', image='Ubuntu 16.04 Xenial Xerus',
        flavor=dict(id='0dab10b5-42a2-438e-be7b-505741a7ffcc'),
        wait=True, auto_ip=True)

    print("\n\nFull Server\n\n")
    cloud.pprint(server)

    print("\n\nTurn Detailed Off\n\n")
    cloud.pprint(cloud.get_server('my-server', detailed=False))

    print("\n\nBare Server\n\n")
    cloud.pprint(cloud.get_server('my-server', bare=True))

finally:
    # Delete it - this is a demo
    cloud.delete_server(server, wait=True, delete_ips=True)
```

Exceptions

- All openstacksdk exceptions are subclasses of *OpenStackCloudException*
- Direct REST calls throw *OpenStackCloudHTTPError*
- *OpenStackCloudHTTPError* subclasses *OpenStackCloudException* and *requests.exceptions.HTTPError*
- *OpenStackCloudURINotFound* for 404
- *OpenStackCloudBadRequest* for 400

User Agent Info

- Set *app_name* and *app_version* for User Agents
- (ssh *region_name* is optional if the cloud has one region)
- examples/cloud/user-agent.py

```
from openstack import cloud as openstack
openstack.enable_logging(http_debug=True)

cloud = openstack.connect(
    cloud='datacentred',
    app_name='AmazingApp',
    app_version='1.0',
```

(continues on next page)

(continued from previous page)

```
)
cloud.list_networks()
```

Uploading Large Objects

- swift has a maximum object size
- Large Objects are uploaded specially
- openstacksdk figures this out and does it
- multi-threaded
- examples/cloud/upload-object.py

```
from openstack import cloud as openstack
openstack.enable_logging(debug=True)

cloud = openstack.connect(cloud='ovh', region_name='SBG1')
cloud.create_object(
    container='my-container',
    name='my-object',
    filename='/home/mordred/briarcliff.sh3d',
)
cloud.delete_object('my-container', 'my-object')
cloud.delete_container('my-container')
```

Uploading Large Objects

- Default max_file_size is 5G
- This is a conference demo
- Lets force a segment_size
- One MILLION bytes
- examples/cloud/upload-object.py

```
from openstack import cloud as openstack
openstack.enable_logging(debug=True)

cloud = openstack.connect(cloud='ovh', region_name='SBG1')
cloud.create_object(
    container='my-container',
    name='my-object',
    filename='/home/mordred/briarcliff.sh3d',
    segment_size=1000000,
)
cloud.delete_object('my-container', 'my-object')
cloud.delete_container('my-container')
```

Service Conditionals

```
from openstack import cloud as openstack
openstack.enable_logging(debug=True)

cloud = openstack.connect(cloud='kiss', region_name='region1')
print(cloud.has_service('network'))
print(cloud.has_service('container-orchestration'))
```

Service Conditional Overrides

- Sometimes clouds are weird and figuring that out wont work

```
from openstack import cloud as openstack
openstack.enable_logging(debug=True)

cloud = openstack.connect(cloud='rax', region_name='DFW')
print(cloud.has_service('network'))
```

```
clouds:
  rax:
    profile: rackspace
    auth:
      username: mordred
      project_id: 245018
    # This is already in profile: rackspace
    has_network: false
```

FIN

CHAPTER

THREE

FOR CONTRIBUTORS

3.1 Contributing to the OpenStack SDK

This section of documentation pertains to those who wish to contribute to the development of this SDK. If youre looking for documentation on how to use the SDK to build applications, refer to the user section.

3.1.1 About the Project

The OpenStack SDK is a OpenStack project aimed at providing a complete software development kit for the programs which make up the OpenStack community. It is a Python library with corresponding documentation, examples, and tools released under the Apache 2 license.

A Brief History

openstacksdk started its life as three different libraries: *shade*, *os-client-config* and *python-openstacksdk*.

shade
: *shade* started its life as some code inside of OpenStack Infras nodepool project, and as some code inside of the Ansible OpenStack Modules. Ansible had a bunch of different OpenStack related modules, and there was a ton of duplicated code. Eventually, between refactoring that duplication into an internal library, and adding the logic and features that the OpenStack Infra team had developed to run client applications at scale, it turned out that wed written nine-tenths of what wed need to have a standalone library.

  Because of its background from nodepool, *shade* contained abstractions to work around deployment differences and is resource oriented rather than service oriented. This allows a user to think about Security Groups without having to know whether Security Groups are provided by Nova or Neutron on a given cloud. On the other hand, as an interface that provides an abstraction, it deviates from the published OpenStack REST API and adds its own opinions, which may not get in the way of more advanced users with specific needs.

os-client-config
: *os-client-config* was a library for collecting client configuration for using an OpenStack cloud in a consistent and comprehensive manner, which introduced the `clouds.yaml` file for expressing named cloud configurations.

python-openstacksdk
: *python-openstacksdk* was a library that exposed the OpenStack APIs to developers in a consistent and predictable manner.

After a while it became clear that there was value in both the high-level layer that contains additional business logic and the lower-level SDK that exposes services and their resources faithfully and consistently as Python objects. Even with both of those layers, it is still beneficial at times to be able to make

direct REST calls and to do so with the same properly configured Session from python-requests. This led to the merge of the three projects.

The original contents of the *shade* library have been moved into `openstack.cloud` and *os-client-config* has been moved in to `openstack.config`.

3.1.2 Contribution Mechanics

Contributing to openstacksdk

If youre interested in contributing to the openstacksdk project, the following will help get you started.

Contributor License Agreement

In order to contribute to the openstacksdk project, you need to have signed OpenStacks contributors agreement.

Please read DeveloperWorkflow before sending your first patch for review. Pull requests submitted through GitHub will be ignored.

See also

- https://wiki.openstack.org/wiki/How_To_Contribute
- https://wiki.openstack.org/wiki/CLA

Project Hosting Details

Project Documentation
https://docs.openstack.org/openstacksdk/latest/

Bug tracker
https://bugs.launchpad.net/openstacksdk

Mailing list (prefix subjects with `[sdk]` for faster responses)
https://lists.openstack.org/mailman3/lists/openstack-discuss.lists.openstack.org/

Code Hosting
https://opendev.org/openstack/openstacksdk

Code Review
https://review.opendev.org/#/q/status:open+project:openstack/openstacksdk,n,z

3.1.3 Contacting the Developers

IRC

The developers of this project are available in the #openstack-sdks channel on OFTC IRC. This channel includes conversation on SDKs and tools within the general OpenStack community, including OpenStackClient as well as occasional talk about SDKs created for languages outside of Python.

Email

The openstack-discuss mailing list fields questions of all types on OpenStack. Using the `[sdk]` filter to begin your email subject will ensure that the message gets to SDK developers.

3.1.4 Coding Standards

We are a bit stricter than usual in the coding standards department. Its a good idea to read through the *coding* section.

OpenStack SDK Developer Coding Standards

In the beginning, there were no guidelines. And it was good. But that didnt last long. As more and more people added more and more code, we realized that we needed a set of coding standards to make sure that the *openstacksdk* API at least *attempted* to display some form of consistency.

Thus, these coding standards/guidelines were developed. Note that not all of *openstacksdk* adheres to these standards just yet. Some older code has not been updated because we need to maintain backward compatibility. Some of it just hasnt been changed yet. But be clear, all new code *must* adhere to these guidelines.

Below are the patterns that we expect *openstacksdk* developers to follow.

Release Notes

openstacksdk uses reno for managing its release notes. A new release note should be added to your contribution anytime you add new API calls, fix significant bugs, add new functionality or parameters to existing API calls, or make any other significant changes to the code base that we should draw attention to for the user base.

It is *not* necessary to add release notes for minor fixes, such as correction of documentation typos, minor code cleanup or reorganization, or any other change that a user would not notice through normal usage.

Exceptions

Exceptions should NEVER be wrapped and re-raised inside of a new exception. This removes important debug information from the user. All of the exceptions should be raised correctly the first time.

openstack.cloud API Methods

The `openstack.cloud` layer has some specific rules:

- When an API call acts on a resource that has both a unique ID and a name, that API call should accept either identifier with a name_or_id parameter.
- All resources should adhere to the get/list/search interface that control retrieval of those resources. E.g., `get_image()`, `list_images()`, `search_images()`.
- Resources should have `create_RESOURCE()`, `delete_RESOURCE()`, `update_RESOURCE()` API methods (as it makes sense).
- For those methods that should behave differently for omitted or None-valued parameters, use the `_utils.valid_kwargs` decorator. This includes all Neutron `update_*` functions.
- Deleting a resource should return True if the delete succeeded, or False if the resource was not found.

Returned Resources

The `openstack.cloud` layer should rely on the proxy layer for the given service. This will ensure complex objects returned to the caller are of `openstack.resource.Resource` type.

Nova vs. Neutron

- Recognize that not all cloud providers support Neutron, so never assume it will be present. If a task can be handled by either Neutron or Nova, code it to be handled by either.
- For methods that accept either a Nova pool or Neutron network, the parameter should just refer to the network, but documentation of it should explain about the pool. See: `create_floating_ip()` and `available_floating_ip()` methods.

Tests

- New API methods *must* have unit tests!
- New unit tests should only mock at the REST layer using `requests_mock`. Any mocking of *openstacksdk* itself should be considered legacy and to be avoided. Exceptions to this rule can be made when attempting to test the internals of a logical shim where the inputs and output of the method arent actually impacted by remote content.
- Functional tests should be added, when possible.
- In functional tests, always use unique names (for resources that have this attribute) and use it for clean up (see next point).
- In functional tests, always define cleanup functions to delete data added by your test, should something go wrong. Data removal should be wrapped in a try except block and try to delete as many entries added by the test as possible.

3.1.5 Development Environment

The first step towards contributing code and documentation is to setup your development environment. We use a pretty standard setup, but it is fully documented in our *setup* section.

Creating a Development Environment

Required Tools

Python

As the OpenStack SDK is developed in Python, you will need at least one version of Python installed. Our continuous integration system runs against several versions, so ultimately we will have the proper test coverage, but having multiple versions locally results in less time spent in code review when changes unexpectedly break other versions.

Python can be downloaded from https://www.python.org/downloads.

virtualenv

In order to isolate our development environment from the system-based Python installation, we use virtualenv. This allows us to install all of our necessary dependencies without interfering with anything

else, and preventing others from interfering with us. Virtualenv must be installed on your system in order to use it, and it can be had from PyPI, via pip, as follows. Note that you may need to run this as an administrator in some situations.:

```
$ apt-get install python3-virtualenv  # Debian based platforms
$ dnf install python3-virtualenv      # Red Hat based platforms
$ pip install virtualenv              # Mac OS X and other platforms
```

You can create a virtualenv in any location. A common usage is to store all of your virtualenvs in the same place, such as under your home directory. To create a virtualenv for the default Python, run the following:

```
$ virtualenv $HOME/envs/sdk
```

To create an environment for a different version, run the following:

```
$ virtualenv -p python3 $HOME/envs/sdk3
```

When you want to enable your environment so that you can develop inside of it, you *activate* it. To activate an environment, run the /bin/activate script inside of it, like the following:

```
$ source $HOME/envs/sdk3/bin/activate
(sdk3)$
```

Once you are activated, you will see the environment name in front of your command prompt. In order to exit that environment, run the `deactivate` command.

tox

We use tox as our test runner, which allows us to run the same test commands against multiple versions of Python. Inside any of the virtualenvs you use for working on the SDK, run the following to install `tox` into it.:

```
(sdk3)$ pip install tox
```

Git

The source of the OpenStack SDK is stored in Git. In order to work with our source repository, you must have Git installed on your system. If your system has a package manager, it can likely be had from there. If not, you can find downloads or the source at http://git-scm.com.

Getting the Source Code

Note

Before checking out the code, please read the OpenStack Developers Guide for details on how to use the continuous integration and code review systems that we use.

The canonical Git repository is hosted on opendev.org at http://opendev.org/openstack/openstacksdk/:

```
(sdk3)$ git clone https://opendev.org/openstack/openstacksdk
(sdk3)$ cd openstacksdk
```

Installing Dependencies

In order to work with the SDK locally, such as in the interactive interpreter or to run example scripts, you need to install the projects dependencies.:

```
(sdk3)$ pip install -r requirements.txt
```

After the downloads and installs are complete, youll have a fully functional environment to use the SDK in.

Building the Documentation

Our documentation is written in reStructured Text and is built using Sphinx. A `docs` command is available in our `tox.ini`, allowing you to build the documentation like youd run tests. The `docs` command is not evaluated by default.:

```
(sdk3)$ tox -e docs
```

That command will cause the documentation, which lives in the `docs` folder, to be built. HTML output is the most commonly referenced, which is located in `docs/build/html`.

3.1.6 Testing

The project contains two test packages, one for unit tests and one for functional tests. The `openstack.tests.unit` package tests the SDKs features in isolation. The `openstack.tests.functional` package tests the SDKs features and examples against an OpenStack cloud.

Testing

The tests are run with tox and configured in `tox.ini`. The test results are tracked by stestr and configured in `.stestr.conf` and via command line options passed to the `stestr` executable when its called by `tox`.

Unit Tests

Running tests

In order to run the entire unit test suite, simply run the `tox` command inside of your source checkout. This will attempt to run every test command listed inside of `tox.ini`, which includes Python 3.x, and a PEP 8 check. You should run the full test suite on all versions before submitting changes for review in order to avoid unexpected failures in the continuous integration system.:

```
$ tox
...
py3: commands succeeded
pep8: commands succeeded
congratulations :)
```

During development, it may be more convenient to run a subset of the tests to keep test time to a minimum. You can choose to run the tests only on one version. A step further is to run only the tests you are working on.:

```
# Run run the tests on Python 3.9
$ tox -e py39
# Run only the compute unit tests on Python 3.9
$ tox -e py39 openstack.tests.unit.compute
# Run only the tests in a specific file on Python 3.9
$ tox -e py39 -- -n openstack/tests/unit/compute/test_version.py
```

Functional Tests

The functional tests assume that you have a public or private OpenStack cloud that you can run the tests against. The tests must be able to be run against public clouds but first and foremost they must be run against OpenStack. In practice, this means that the tests should initially be run against a stable branch of DevStack.

Configuration

To connect the functional tests to an OpenStack cloud we require a `clouds.yaml` file, as discussed in *Configuring OpenStack SDK Applications*. You can place this `clouds.yaml` file in the root of your source checkout or in one of the other standard locations, `$HOME/.config/openstack` or `/etc/openstack`.

There must be at least three clouds configured, or rather three accounts configured for the one cloud. These accounts are:

- An admin account, which defaults to `devstack-admin` but is configurable via the `OPENSTACKSDK_OPERATOR_CLOUD` environment variable,
- A user account, which defaults to `devstack` but is configurable via the `OPENSTACKSDK_DEMO_CLOUD` environment variable, and
- An alternate user account, which defaults to `devstack-demo` but is configurable via the `OPENSTACKSDK_DEMO_CLOUD_ALT` environment variable

In addition, you must indicate the names of the flavor and image that should be used for tests. These can be configured via `OPENSTACKSDK_FLAVOR` and `OPENSTACKSDK_IMAGE` environment variables or `functional.flavor_name` and `functional.image_name` settings in the `clouds.yaml` file, respectively.

Finally, you can configure the timeout for tests using the `OPENSTACKSDK_FUNC_TEST_TIMEOUT` environment variable (defaults to 300 seconds). Some test modules take specific timeout values. For example, all tests in `openstack.tests.functional.compute` will check for the `OPENSTACKSDK_FUNC_TEST_TIMEOUT_COMPUTE` environment variable before checking for `OPENSTACKSDK_FUNC_TEST_TIMEOUT`.

Note

Recent versions of DevStack will configure a suitable `clouds.yaml` file for you, which will be placed at `/etc/openstack/clouds.yaml`.

This is an example of a minimal configuration for a `clouds.yaml` that connects the functional tests to a DevStack instance.

```
clouds:
  devstack:
    auth:
      auth_url: http://xxx.xxx.xxx.xxx/identity
      password: password
      project_domain_id: default
      project_name: demo
      user_domain_id: default
      username: demo
    identity_api_version: '3'
    region_name: RegionOne
    volume_api_version: '3'
  devstack-admin:
    auth:
      auth_url: http://xxx.xxx.xxx.xxx/identity
      password: password
      project_domain_id: default
      project_name: admin
      user_domain_id: default
      username: admin
    identity_api_version: '3'
    region_name: RegionOne
    volume_api_version: '3'
  devstack-alt:
    auth:
      auth_url: http://xxx.xxx.xxx.xxx/identity
      password: password
      project_domain_id: default
      project_name: alt_demo
      user_domain_id: default
      username: alt_demo
    identity_api_version: '3'
    region_name: RegionOne
    volume_api_version: '3'
example:
  image_name: cirros-0.5.2-x86_64-disk
  flavor_name: m1.small
```

Replace `xxx.xxx.xxx.xxx` with the IP address or FQDN of your DevStack instance.

Running tests

Functional tests are also run against multiple Python versions. In order to run the entire functional test suite against the default Python 3 version in your environment, run the `tox -e functional` command inside of your source checkout. This will attempt to run every tests in the `openstack/tests/functional` directory. For example:

```
$ tox -e functional
...
functional: commands succeeded
congratulations :)
```

3.1.7 Project Layout

The project contains a top-level `openstack` package, which houses several modules that form the foundation upon which each services API is built on. Under the `openstack` package are packages for each of those services, such as `openstack.compute`.

How the SDK is organized

The following diagram shows how the project is laid out.

```
openstack/
    connection.py
    resource.py
    compute/
        compute_service.py
        v2/
            server.py
            _proxy.py
    tests/
        compute/
            v2/
                test_server.py
```

Resource

The *openstack.resource.Resource* base class is the building block of any service implementation. `Resource` objects correspond to the resources each services REST API works with, so the *openstack.compute.v2.server.Server* subclass maps to the compute services `https://openstack:1234/v2/servers` resource.

The base `Resource` contains methods to support the typical CRUD operations supported by REST APIs, and handles the construction of URLs and calling the appropriate HTTP verb on the given `Adapter`.

Values sent to or returned from the service are implemented as attributes on the `Resource` subclass with type `openstack.resource.prop`. The `prop` is created with the exact name of what the API expects, and can optionally include a `type` to be validated against on requests. You should choose an attribute name that follows PEP-8, regardless of what the server-side expects, as this `prop` becomes a mapping between the two.:

```
is_public = resource.prop('os-flavor-access:is_public', type=bool)
```

There are six additional attributes which the `Resource` class checks before making requests to the REST API. `allow_create`, `allow_retreive`, `allow_commit`, `allow_delete`, `allow_head`, and `allow_list` are set to `True` or `False`, and are checked before making the corresponding method call.

The `base_path` attribute should be set to the URL which corresponds to this resource. Many `base_path`s are simple, such as `"/servers"`. For `base_path`s which are composed of non-static information, Pythons string replacement is used, e.g., `base_path = "/servers/%(server_id)s/ips"`.

resource_key and resources_key are attributes to set when a Resource returns more than one item in a response, or otherwise requires a key to obtain the response value. For example, the Server class sets resource_key = "server" as an individual Server is stored in a dictionary keyed with the singular noun, and resources_key = "servers" as multiple Servers are stored in a dictionary keyed with the plural noun in the response.

Proxy

Each service implements a Proxy class based on *Proxy*, within the openstack/<program_name>/vX/_proxy.py module. For example, the v2 compute services Proxy exists in openstack/compute/v2/_proxy.py.

The *Proxy* class is based on Adapter.

class openstack.proxy.**Proxy**(*session*, *statsd_client=None*, *statsd_prefix=None*, *prometheus_counter=None*, *prometheus_histogram=None*, *influxdb_config=None*, *influxdb_client=None*, **args*, ***kwargs*)

Bases: Adapter

Represents a service.

retriable_status_codes = None

HTTP status codes that should be retried by default.

The number of retries is defined by the configuration in parameters called <service-type>_status_code_retries.

Each services Proxy provides a higher-level interface for users to work with via a *Connection* instance.

Rather than requiring users to maintain their own Adapter and work with lower-level *Resource* objects, the Proxy interface offers a place to make things easier for the caller.

Each Proxy class implements methods which act on the underlying Resource classes which represent the service. For example:

```
def list_flavors(self, **params):
    return flavor.Flavor.list(self.session, **params)
```

This method is operating on the openstack.compute.v2.flavor.Flavor.list method. For the time being, it simply passes on the Adapter maintained by the Proxy, and returns what the underlying Resource.list method does.

Cloud

TODO.

Connection

The *openstack.connection.Connection* class builds atop a *openstack.config.cloud_region.CloudRegion* object, and provides a higher level interface constructed of Proxy objects from each of the services.

The Connection class primary purpose is to act as a high-level interface to this SDK, managing the lower level connection bits and exposing the Resource objects through their corresponding *Proxy* object.

If youve built proper Resource objects and implemented methods on the corresponding Proxy object, the high-level interface to your service should now be exposed.

3.1.8 Adding Features

Does this SDK not do what you need it to do? Is it missing a service? Are you a developer on another project who wants to add their service? Youre in the right place. Below are examples of how to add new features to the OpenStack SDK.

Creating a New Resource

This guide will walk you through how to add resources for a service.

Naming Conventions

Above all, names across this project conform to Pythons naming standards, as laid out in PEP 8.

The relevant details we need to know are as follows:

- Module names are lower case, and separated by underscores if more than one word. For example, `openstack.object_store`
- Class names are capitalized, with no spacing, and each subsequent word is capitalized in a name. For example, `ServerMetadata`.
- Attributes on classes, including methods, are lower case and separated by underscores. For example, `allow_list` or `get_data`.

Services

Services in the OpenStack SDK are named after their program name, not their code name. For example, the project often known as Nova is always called compute within this SDK.

This guide walks through creating service for an OpenStack program called Fake. Following our guidelines, the code for its service would live under the `openstack.fake` namespace. What follows is the creation of a *`Resource`* class for the Fake service.

Resources

Resources are named after the server-side resource, which is set in the `base_path` attribute of the resource class. This guide creates a resource class for the `/fake` server resource, so the resource module is called `fake.py` and the class is called `Fake`.

An Example

`openstack/fake/fake_service.py`

```
# Apache 2 header omitted for brevity

from openstack import service_description
from openstack.fake.v2 import _proxy as _proxy_v2


class FakeService(service_description.ServiceDescription):
    """The fake service."""

    supported_versions = {
```

(continues on next page)

(continued from previous page)

```
        '2': _proxy_v2.Proxy,
    }
```

openstack/fake/v2/fake.py

```
# Apache 2 header omitted for brevity

from openstack import resource


class Fake(resource.Resource):
    resource_key = "resource"
    resources_key = "resources"
    base_path = "/fake"

    allow_create = True
    allow_fetch = True
    allow_commit = True
    allow_delete = True
    allow_list = True
    allow_head = True

    #: The transaction date and time.
    timestamp = resource.Header("x-timestamp")
    #: The name of this resource.
    name = resource.Body("name", alternate_id=True)
    #: The value of the resource. Also available in headers.
    value = resource.Body("value", alias="x-resource-value")
    #: Is this resource cool? If so, set it to True.
    #: This is a multi-line comment about cool stuff.
    cool = resource.Body("cool", type=bool)
```

`fake.Fake` Attributes

Each services resources inherit from *`Resource`*, so they can override any of the base attributes to fit the way their particular resource operates.

`resource_key` and `resources_key`

These attributes are set based on how your resource responds with data. The default values for each of these are `None`, which works fine when your resource returns a JSON body that can be used directly without a top-level key, such as `{"name": "Ernie Banks", ...}`.

However, our `Fake` resource returns JSON bodies that have the details of the resource one level deeper, such as `{"resources": {"name": "Ernie Banks", ...}, {...}}`. It does a similar thing with single resources, putting them inside a dictionary keyed on `"resource"`.

By setting `Fake.resource_key` on *line 8*, we tell the `Resource.create`, `Resource.get`, and `Resource.update` methods that were either sending or receiving a resource that is in a dictionary with that key.

By setting `Fake.resources_key` on *line 9*, we tell the `Resource.list` method that were expecting to receive multiple resources inside a dictionary with that key.

base_path

The `base_path` is the URL were going to use to make requests for this resource. In this case, *line 10* sets `base_path = "/fake"`, which also corresponds to the name of our class, `Fake`.

Most resources follow this basic formula. Some cases are more complex, where the URL to make requests to has to contain some extra data. The volume service has several resources which make either basic requests or detailed requests, so they use `base_path = "/volumes/%s(detailed)"`. Before a request is made, if `detailed = True`, they convert it to a string so the URL becomes `/volumes/detailed`. If its `False`, they only send `/volumes/`.

service

Line 11 is an instance of the service were implementing. Each resource ties itself to the service through this setting, so that the proper URL can be constructed.

In `fake_service.py`, we specify the valid versions as well as what this service is called in the service catalog. When a request is made for this resource, the Session now knows how to construct the appropriate URL using this `FakeService` instance.

Supported Operations

The base *Resource* disallows all types of requests by default, requiring each resource to specify which requests they support. On *lines 14-19*, our `Fake` resource specifies that itll work with all of the operations.

In order to have the following methods work, you must allow the corresponding value by setting it to `True`:

create	allow_create
delete	allow_delete
head	allow_head
list	allow_list
fetch	allow_fetch
commit	allow_commit

An additional attribute to set is `commit_method`. It defaults to `PUT`, but some services use `POST` or `PATCH` to commit changes back to the remote resource.

Properties

The way resource classes communicate values between the user and the server are `prop` objects. These act similarly to Pythons built-in property objects, but they share only the name - theyre not the same.

Properties are set based on the contents of a response body or headers. Based on what your resource returns, you should set `prop`s to map those values to ones on your *Resource* object.

Line 22 sets a prop for `timestamp` , which will cause the `Fake.timestamp` attribute to contain the value returned in an `X-Timestamp` header, such as from a `Fake.head` request.

Line 24 sets a prop for `name`, which is a value returned in a body, such as from a `Fake.get` request. Note from *line 12* that `name` is specified its `id` attribute, so when this resource is populated from a response, `Fake.name` and `Fake.id` are the same value.

Line 26 sets a prop which contains an alias. `Fake.value` will be set when a response body contains a `value`, or when a header contains `X-Resource-Value`.

Line 28 specifies a type to be checked before sending the value in a request. In this case, we can only set `Fake.cool` to either `True` or `False`, otherwise a TypeError will be raised if the value cant be converted to the expected type.

Documentation

We use Sphinxs `autodoc` feature in order to build API documentation for each resource we expose. The attributes we override from *`Resource`* dont need to be documented, but any `prop` attributes must be. All you need to do is add a comment *above* the line to document, with a colon following the pound-sign.

Lines 21, 23, 25, and 27-28 are comments which will then appear in the API documentation. As shown in *lines 27 & 28*, these comments can span multiple lines.

CHAPTER

FOUR

GENERAL INFORMATION

General information about the SDK including a glossary and release history.

4.1 Glossary

CLI
Command-Line Interface; a textual user interface.

compute
OpenStack Compute (Nova).

container
One of the *object-store* resources; a container holds *objects* being stored.

endpoint
A base URL used in a REST request. An *authentication endpoint* is specifically the URL given to a user to identify a cloud. A service endpoint is generally obtained from the service catalog.

host
A physical computer. Contrast with *node* and *server*.

identity
OpenStack Identity (Keystone).

image
OpenStack Image (Glance). Also the attribute name of the disk files stored for use by servers.

keypair
The attribute name of the SSH public key used in the OpenStack Compute API for server authentication.

node
A logical system, may refer to a *server* (virtual machine) or a *host*.

Generally used to describe an OS instance where a specific process is running, e.g. a network node is where the network processes run, and may be directly on a host or in a server. Contrast with *host* and *server*.

object
A generic term which normally refers to the a Python `object`. The OpenStack Object Store service (Swift) also uses *object* as the name of the item being stored within a *container*.

object-store
OpenStack Object Store (Swift).

project
: The name of the owner of resources in an OpenStack cloud. A *project* can map to a customer, account or organization in different OpenStack deployments. Used instead of the deprecated *tenant*.

region
: The attribute name of a partitioning of cloud resources.

resource
: A Python object representing an OpenStack resource inside the SDK code. Also used to describe the items managed by OpenStack.

role
: A personality that a user assumes when performing a specific set of operations. A *role* includes a set of rights and privileges that a user assuming that role inherits. The OpenStack Identity service includes the set of roles that a user can assume in the *token* that is issued to that user.

  The individual services determine how the roles are interpreted and access granted to operations or resources. The OpenStack Identity service treats a role as an arbitrary name assigned by the cloud administrator.

server
: A virtual machine or a bare-metal host managed by the OpenStack Compute service. Contrast with *host* and *node*.

service
: In OpenStack this refers to a service/endpoint in the *ServiceCatalog*. It could also be a collection of endpoints for different *regions*. A service has a type and a name.

service catalog
: The list of *services* configured at a given authentication endpoint available to the authenticated user.

tenant
: Deprecated in favor of *project*.

token
: An arbitrary bit of text that is used to access resources. Some tokens are *scoped* to determine what resources are accessible with it. A token may be revoked at any time and is valid for a finite duration.

volume
: OpenStack Volume (Cinder). Also the attribute name of the virtual disks managed by the OpenStack Volume service.

4.2 Release Notes

Release notes for *openstacksdk* can be found at https://releases.openstack.org/teams/openstacksdk.html

PYTHON MODULE INDEX

O

INDEX

A

B

C

D

E

F

G

H

I

M

N

O

P

Q

R

S

T

U

V

W

Z