python-keystoneclient Documentation

Release 5.0.2.dev2

OpenStack

Sep 13, 2022

CONTENTS

1 Using the V3 Client API **3**
- 1.1 Introduction . . . 3
- 1.2 Authenticating Using Sessions . . . 4
- 1.3 Getting Metadata Responses . . . 5
- 1.4 Non-Session Authentication (deprecated) . . . 5

2 Using Sessions **7**
- 2.1 Introduction . . . 7
  - 2.1.1 Features . . . 7
- 2.2 Sessions for Users . . . 7
  - 2.2.1 Migrating keystoneclient to use a Session . . . 8
  - 2.2.2 Sharing Authentication Plugins . . . 8
- 2.3 Sessions for Client Developers . . . 9
  - 2.3.1 Authentication . . . 9
  - 2.3.2 Service Discovery . . . 9

3 Using the V2 client API **11**
- 3.1 Introduction . . . 11
- 3.2 Authenticating . . . 11
- 3.3 Creating tenants . . . 12
- 3.4 Creating users . . . 12
- 3.5 Creating roles and adding users . . . 13
- 3.6 Creating services and endpoints . . . 13

4 keystoneclient **15**
- 4.1 keystoneclient package . . . 15
  - 4.1.1 Subpackages . . . 15
    - keystoneclient.auth package . . . 15
    - keystoneclient.common package . . . 65
    - keystoneclient.contrib package . . . 68
    - keystoneclient.generic package . . . 73
    - keystoneclient.v2_0 package . . . 74
    - keystoneclient.v3 package . . . 83
  - 4.1.2 Submodules . . . 147
  - 4.1.3 keystoneclient.access module . . . 147
  - 4.1.4 keystoneclient.adapter module . . . 159
  - 4.1.5 keystoneclient.base module . . . 162
  - 4.1.6 keystoneclient.baseclient module . . . 164
  - 4.1.7 keystoneclient.client module . . . 165

4.1.8 keystoneclient.discover module . . . 166
4.1.9 keystoneclient.exceptions module . . . 170
4.1.10 keystoneclient.httpclient module . . . 180
4.1.11 keystoneclient.i18n module . . . 188
4.1.12 keystoneclient.service_catalog module . . . 188
4.1.13 keystoneclient.session module . . . 192
4.1.14 keystoneclient.utils module . . . 200
4.1.15 Module contents . . . 200

5 Related Identity Projects **201**

6 Release Notes **203**

7 Contributing **205**

8 Indices and tables **207**

This is a client for OpenStack Identity API. Theres a Python API for *Identity API v3* and *v2* (the `keystoneclient` modules).

Contents:

CHAPTER

ONE

USING THE V3 CLIENT API

1.1 Introduction

The main concepts in the Identity v3 API are:

- *credentials*
- *domain_configs*
- *domains*
- *endpoints*
- *groups*
- *policies*
- *projects*
- *regions*
- *role_assignments*
- *roles*
- *services*
- *tokens*
- *users*

The *keystoneclient.v3.client* API lets you query and make changes through `managers`. For example, to manipulate a project (formerly called tenant), you interact with a *keystoneclient.v3.projects.ProjectManager* object.

You obtain access to managers through attributes of a *keystoneclient.v3.client.Client* object. For example, the `projects` attribute of a `Client` object is a projects manager:

```
>>> from keystoneclient.v3 import client
>>> keystone = client.Client(...)
>>> keystone.projects.list() # List projects
```

While it is possible to instantiate a *keystoneclient.v3.client.Client* object (as done above for clarity), the recommended approach is to use the discovery mechanism provided by the *keystoneclient.client.Client* class. The appropriate class will be instantiated depending on the API versions available:

```
>>> from keystoneclient import client
>>> keystone =
...     client.Client(auth_url='http://localhost:5000', ...)
>>> type(keystone)
<class 'keystoneclient.v3.client.Client'>
```

One can force the use of a specific version of the API, either by using the `version` keyword argument:

```
>>> from keystoneclient import client
>>> keystone = client.Client(auth_url='http://localhost:5000',
                             version=(2,), ...)
>>> type(keystone)
<class 'keystoneclient.v2_0.client.Client'>
>>> keystone = client.Client(auth_url='http://localhost:5000',
                             version=(3,), ...)
>>> type(keystone)
<class 'keystoneclient.v3.client.Client'>
```

Or by specifying directly the specific API version authentication URL as the auth_url keyword argument:

```
>>> from keystoneclient import client
>>> keystone =
...     client.Client(auth_url='http://localhost:5000/v2.0', ...)
>>> type(keystone)
<class 'keystoneclient.v2_0.client.Client'>
>>> keystone =
...     client.Client(auth_url='http://localhost:5000/v3', ...)
>>> type(keystone)
<class 'keystoneclient.v3.client.Client'>
```

Upon successful authentication, a *keystoneclient.v3.client.Client* object is returned (when using the Identity v3 API). Authentication and examples of common tasks are provided below.

You can generally expect that when the client needs to propagate an exception it will raise an instance of subclass of *keystoneclient.exceptions.ClientException*.

1.2 Authenticating Using Sessions

Instantiate a *keystoneclient.v3.client.Client* using a `Session` to provide the authentication plugin, SSL/TLS certificates, and other data:

```
>>> from keystoneauth1.identity import v3
>>> from keystoneauth1 import session
>>> from keystoneclient.v3 import client
>>> auth = v3.Password(auth_url='https://my.keystone.com:5000/v3',
...                    user_id='myuserid',
...                    password='mypassword',
...                    project_id='myprojectid')
>>> sess = session.Session(auth=auth)
>>> keystone = client.Client(session=sess)
```

For more information on Sessions refer to: Using Sessions.

1.3 Getting Metadata Responses

Instantiating *keystoneclient.v3.client.Client* using *include_metadata=True* will cause manager response to return *keystoneclient.base.Response* instead of just the data. The metadata property will be available directly to the *keystoneclient.base.Response* and the response data will be available as property *data* to it.

```
>>> from keystoneauth1.identity import v3
>>> from keystoneauth1 import session
>>> from keystoneclient.v3 import client
>>> auth = v3.Password(auth_url='https://my.keystone.com:5000/v3',
...                    user_id='myuserid',
...                    password='mypassword',
...                    project_id='myprojectid')
>>> sess = session.Session(auth=auth)
>>> keystone = client.Client(session=sess, include_metadata=True)
>>> resp = keystone.projects.list()
>>> resp.request_ids[0]
req-1234-5678-...
>>> resp.data
[<Project ...>, <Project ...>, ...]
```

1.4 Non-Session Authentication (deprecated)

The *deprecated* way to authenticate is to pass the username, the users domain name (which will default to Default if it is not specified), and a password:

```
>>> from keystoneclient import client
>>> auth_url = 'http://localhost:5000'
>>> username = 'adminUser'
>>> user_domain_name = 'Default'
>>> password = 'secreetword'
>>> keystone = client.Client(auth_url=auth_url, version=(3,),
...                          username=username, password=password,
...                          user_domain_name=user_domain_name)
```

A `Session` should be passed to the Client instead. Using a Session youre not limited to authentication using a username and password but can take advantage of other more secure authentication methods.

You may optionally specify a domain or project (along with its project domain name), to obtain a scoped token:

```
>>> from keystoneclient import client
>>> auth_url = 'http://localhost:5000'
>>> username = 'adminUser'
>>> user_domain_name = 'Default'
>>> project_name = 'demo'
```

(continues on next page)

(continued from previous page)

```
>>> project_domain_name = 'Default'
>>> password = 'secreetword'
>>> keystone = client.Client(auth_url=auth_url, version=(3,),
...                          username=username, password=password,
...                          user_domain_name=user_domain_name,
...                          project_name=project_name,
...                          project_domain_name=project_domain_name)
```

CHAPTER

TWO

USING SESSIONS

2.1 Introduction

The `keystoneauth1.session.Session` class was introduced into keystoneclient as an attempt to bring a unified interface to the various OpenStack clients that share common authentication and request parameters between a variety of services.

The model for using a Session and auth plugin as well as the general terms used have been heavily inspired by the requests library. However neither the Session class nor any of the authentication plugins rely directly on those concepts from the requests library so you should not expect a direct translation.

2.1.1 Features

- Common client authentication

  Authentication is handled by one of a variety of authentication plugins and then this authentication information is shared between all the services that use the same Session object.

- Security maintenance

  Security code is maintained in a single place and reused between all clients such that in the event of problems it can be fixed in a single location.

- Standard discovery mechanisms

  Clients are not expected to have any knowledge of an identity token or any other form of identification credential. Service and endpoint discovery are handled by the Session and plugins.

2.2 Sessions for Users

The Session object is the contact point to your OpenStack cloud services. It stores the authentication credentials and connection information required to communicate with OpenStack such that it can be reused to communicate with many services. When creating services this Session object is passed to the client so that it may use this information.

A Session will authenticate on demand. When a request that requires authentication passes through the Session the authentication plugin will be asked for a valid token. If a valid token is available it will be used otherwise the authentication plugin may attempt to contact the authentication service and fetch a new one.

An example from keystoneclient:

```
>>> from keystoneauth1.identity import v3
>>> from keystoneauth1 import session
>>> from keystoneclient.v3 import client

>>> auth = v3.Password(auth_url='https://my.keystone.com:5000/v3',
...                    username='myuser',
...                    password='mypassword',
...                    project_id='proj',
...                    user_domain_id='domain')
>>> sess = session.Session(auth=auth,
...                        verify='/path/to/ca.cert')
>>> ks = client.Client(session=sess)
>>> users = ks.users.list()
```

As clients adopt this means of operating they will be created in a similar fashion by passing the Session object to the clients constructor.

2.2.1 Migrating keystoneclient to use a Session

By using a session with a keystoneclient Client we presume that you have opted in to new behavior defined by the session. For example authentication is now on-demand rather than on creation. To allow this change in behavior there are a number of functions that have changed behavior or are no longer available.

For example the *keystoneclient.httpclient.HTTPClient.authenticate()* method used to be able to always re-authenticate the current client and fetch a new token. As this is now controlled by the Session and not the client this has changed, however the function will still exist to provide compatibility with older clients.

Likewise certain parameters such as `user_id` and `auth_token` that used to be available on the client object post authentication will remain uninitialized.

When converting an application to use a session object with keystoneclient you should be aware of the possibility of changes to authentication and authentication parameters and make sure to test your code thoroughly. It should have no impact on the typical CRUD interaction with the client.

2.2.2 Sharing Authentication Plugins

A session can only contain one authentication plugin however there is nothing that specifically binds the authentication plugin to that session, a new Session can be created that reuses the existing authentication plugin:

```
>>> new_sess = session.Session(auth=sess.auth,
                               verify='/path/to/different-cas.cert')
```

In this case we cannot know which session object will be used when the plugin performs the authentication call so the command must be able to succeed with either.

Authentication plugins can also be provided on a per-request basis. This will be beneficial in a situation where a single session is juggling multiple authentication credentials:

```
>>> sess.get('https://my.keystone.com:5000/v3',
             auth=my_auth_plugin)
```

If an auth plugin is provided via parameter then it will override any auth plugin on the session.

2.3 Sessions for Client Developers

Sessions are intended to take away much of the hassle of dealing with authentication data and token formats. Clients should be able to specify filter parameters for selecting the endpoint and have the parsing of the catalog managed for them.

2.3.1 Authentication

When making a request with a session object you can simply pass the keyword parameter `authenticated` to indicate whether the argument should contain a token, by default a token is included if an authentication plugin is available:

```
>>> # In keystone this route is unprotected by default
>>> resp = sess.get('https://my.keystone.com:5000/v3',
                    authenticated=False)
```

2.3.2 Service Discovery

In OpenStack the URLs of available services are distributed to the user as a part of the token they receive called the Service Catalog. Clients are expected to use the URLs from the Service Catalog rather than have them provided.

In general a client does not need to know the full URL for the server that they are communicating with, simply that it should send a request to a path belonging to the correct service.

This is controlled by the `endpoint_filter` parameter to a request which contains all the information an authentication plugin requires to determine the correct URL to which to send a request. When using this mode only the path for the request needs to be specified:

```
>>> resp = session.get('/v3/users',
                       endpoint_filter={'service_type': 'identity',
                                        'interface': 'public',
                                        'region_name': 'myregion'})
```

`endpoint_filter` accepts a number of arguments with which it can determine an endpoint url:

- `service_type`: the type of service. For example `identity`, `compute`, `volume` or many other predefined identifiers.
- `interface`: the network exposure the interface has. This will be one of:
  - `public`: An endpoint that is available to the wider internet or network.
  - `internal`: An endpoint that is only accessible within the private network.
  - `admin`: An endpoint to be used for administrative tasks.

- `region_name`: the name of the region where the endpoint resides.

The endpoint filter is a simple key-value filter and can be provided with any number of arguments. It is then up to the auth plugin to correctly use the parameters it understands.

The session object determines the URL matching the filter and append to it the provided path and so create a valid request. If multiple URL matches are found then any one may be chosen.

While authentication plugins will endeavour to maintain a consistent set of arguments for an `endpoint_filter` the concept of an authentication plugin is purposefully generic and a specific mechanism may not know how to interpret certain arguments and ignore them. For example the `keystoneauth1.identity.generic.token.Token` plugin (which is used when you want to always use a specific endpoint and token combination) will always return the same endpoint regardless of the parameters to `endpoint_filter` or a custom OpenStack authentication mechanism may not have the concept of multiple `interface` options and choose to ignore that parameter.

There is some expectation on the user that they understand the limitations of the authentication system they are using.

CHAPTER

THREE

USING THE V2 CLIENT API

3.1 Introduction

The main concepts in the Identity v2 API are:

- tenants
- users
- roles
- services
- endpoints

The V2 client API lets you query and make changes through managers. For example, to manipulate tenants, you interact with a `keystoneclient.v2_0.tenants.TenantManager` object.

You obtain access to managers via attributes of the `keystoneclient.v2_0.client.Client` object. For example, the `tenants` attribute of the `Client` class is a tenant manager:

```
>>> from keystoneclient.v2_0 import client
>>> keystone = client.Client(...)
>>> keystone.tenants.list() # List tenants
```

You create a valid `keystoneclient.v2_0.client.Client` object by passing a `Session` to the constructor. Authentication and examples of common tasks are provided below.

You can generally expect that when the client needs to propagate an exception it will raise an instance of subclass of `keystoneclient.exceptions.ClientException`

3.2 Authenticating

There are two ways to authenticate against keystone:

- against the admin endpoint with the admin token
- against the public endpoint with a username and password

If you are an administrator, you can authenticate by connecting to the admin endpoint and using the admin token (sometimes referred to as the service token). The token is specified as the `admin_token` configuration option in your keystone.conf config file, which is typically in /etc/keystone:

```
>>> from keystoneauth1.identity import v2
>>> from keystoneauth1 import session
>>> from keystoneclient.v2_0 import client
>>> token = '012345SECRET99TOKEN012345'
>>> endpoint = 'http://192.168.206.130:35357/v2.0'
>>> auth = v2.Token(auth_url=endpoint, token=token)
>>> sess = session.Session(auth=auth)
>>> keystone = client.Client(session=sess)
```

If you have a username and password, authentication is done against the public endpoint. You must also specify a tenant that is associated with the user:

```
>>> from keystoneauth1.identity import v2
>>> from keystoneauth1 import session
>>> from keystoneclient.v2_0 import client
>>> username='adminUser'
>>> password='secretword'
>>> tenant_name='openstackDemo'
>>> auth_url='http://192.168.206.130:5000/v2.0'
>>> auth = v2.Password(username=username, password=password,
...                   tenant_name=tenant_name, auth_url=auth_url)
>>> sess = session.Session(auth=auth)
>>> keystone = client.Client(session=sess)
```

3.3 Creating tenants

This example will create a tenant named *openstackDemo*:

```
>>> from keystoneclient.v2_0 import client
>>> keystone = client.Client(...)
>>> keystone.tenants.create(tenant_name="openstackDemo",
...                         description="Default Tenant", enabled=True)
<Tenant {'id': '9b7962da6eb04745b477ae920ad55939', 'enabled': True,
↪'description': 'Default Tenant', 'name': 'openstackDemo'}>
```

3.4 Creating users

This example will create a user named *adminUser* with a password *secretword* in the openstackDemo tenant. We first need to retrieve the tenant:

```
>>> from keystoneclient.v2_0 import client
>>> keystone = client.Client(...)
>>> tenants = keystone.tenants.list()
>>> my_tenant = [x for x in tenants if x.name=='openstackDemo'][0]
>>> my_user = keystone.users.create(name="adminUser",
...                                 password="secretword",
...                                 tenant_id=my_tenant.id)
```

3.5 Creating roles and adding users

This example will create an admin role and add the *my_user* user to that role, but only for the *my_tenant* tenant:

```
>>> from keystoneclient.v2_0 import client
>>> keystone = client.Client(...)
>>> role = keystone.roles.create('admin')
>>> my_tenant = ...
>>> my_user = ...
>>> keystone.roles.add_user_role(my_user, role, my_tenant)
```

3.6 Creating services and endpoints

This example will create the service and corresponding endpoint for the Compute service:

```
>>> from keystoneclient.v2_0 import client
>>> keystone = client.Client(...)
>>> service = keystone.services.create(name="nova", service_type="compute",
...                                    description="Nova Compute Service")
>>> keystone.endpoints.create(
...     region="RegionOne", service_id=service.id,
...     publicurl="http://192.168.206.130:8774/v2/%(tenant_id)s",
...     adminurl="http://192.168.206.130:8774/v2/%(tenant_id)s",
...     internalurl="http://192.168.206.130:8774/v2/%(tenant_id)s")
```

CHAPTER

FOUR

KEYSTONECLIENT

4.1 keystoneclient package

4.1.1 Subpackages

keystoneclient.auth package

Subpackages

keystoneclient.auth.identity package

Subpackages

keystoneclient.auth.identity.generic package

Submodules

keystoneclient.auth.identity.generic.base module

class keystoneclient.auth.identity.generic.base.**BaseGenericPlugin**(*auth_url*, *tenant_id=None*, *tenant_name=None*, *project_id=None*, *project_name=None*, *project_domain_id=None*, *project_domain_name=None*, *domain_id=None*, *domain_name=None*, *trust_id=None*)

Bases: *BaseIdentityPlugin*

An identity plugin that is not version dependent.

Internally we will construct a version dependent plugin with the resolved URL and then proxy all calls from the base plugin to the versioned one.

abstract create_plugin(*session*, *version*, *url*, *raw_status=None*)

Create a plugin from the given parameters.

This function will be called multiple times with the version and url of a potential endpoint. If a plugin can be constructed that fits the params then it should return it. If not return None and then another call will be made with other available URLs.

Parameters

- **session** (keystoneclient.session.Session) A session object.
- **version** (*tuple*) A tuple of the API version at the URL.
- **url** (*string*) The base URL for this version.
- **raw_status** (*string*) The status that was in the discovery field.

Returns
A plugin that can match the parameters or None if nothing.

get_auth_ref(*session*, ***kwargs*)

Obtain a token from an OpenStack Identity Service.

This method is overridden by the various token version plugins.

This method should not be called independently and is expected to be invoked via the do_authenticate() method.

This method will be invoked if the AccessInfo object cached by the plugin is not valid. Thus plugins should always fetch a new AccessInfo when invoked. If you are looking to just retrieve the current auth data then you should use get_access().

Parameters
session (keystoneclient.session.Session) A session object that can be used for communication.

Raises

- *keystoneclient.exceptions.InvalidResponse* The response returned wasnt appropriate.
- *keystoneclient.exceptions.HttpError* An error from an invalid HTTP response.

Returns
Token access information.

Return type
keystoneclient.access.AccessInfo

classmethod get_options()

Return the list of parameters associated with the auth plugin.

This list may be used to generate CLI or config arguments.

Returns
A list of Param objects describing available plugin parameters.

Return type
List

property trust_id

Deprecated as of the 1.7.0 release.

It may be removed in the 2.0.0 release.

keystoneclient.auth.identity.generic.base.**get_options**()

keystoneclient.auth.identity.generic.cli module

class keystoneclient.auth.identity.generic.cli.**DefaultCLI**(*endpoint=None*, *token=None*, ***kwargs*)

Bases: *Password*

A Plugin that provides typical authentication options for CLIs.

This plugin provides standard username and password authentication options as well as allowing users to override with a custom token and endpoint.

get_endpoint(**args*, ***kwargs*)

Return a valid endpoint for a service.

If a valid token is not present then a new one will be fetched using the session and kwargs.

Parameters

- **session** (keystoneclient.session.Session) A session object that can be used for communication.
- **service_type** (*string*) The type of service to lookup the endpoint for. This plugin will return None (failure) if service_type is not provided.
- **interface** (*string*) The exposure of the endpoint. Should be *public*, *internal*, *admin*, or *auth*. *auth* is special here to use the *auth_url* rather than a URL extracted from the service catalog. Defaults to *public*.
- **region_name** (*string*) The region the endpoint should exist in. (optional)
- **service_name** (*string*) The name of the service in the catalog. (optional)
- **version** (*tuple*) The minimum version number required for this endpoint. (optional)

Raises

keystoneclient.exceptions.HttpError An error from an invalid HTTP response.

Returns

A valid endpoint URL or None if not available.

Return type

string or None

classmethod get_options()

Return the list of parameters associated with the auth plugin.

This list may be used to generate CLI or config arguments.

Returns

A list of Param objects describing available plugin parameters.

Return type
List

`get_token`(**args*, ***kwargs*)

Return a valid auth token.

If a valid token is not present then a new one will be fetched.

Parameters
session (keystoneclient.session.Session) A session object that can be used for communication.

Raises
keystoneclient.exceptions.HttpError An error from an invalid HTTP response.

Returns
A valid token.

Return type
string

`classmethod load_from_argparse_arguments`(*namespace*, ***kwargs*)

Load a specific plugin object from an argparse result.

Convert the results of a parse into the specified plugin.

Parameters
namespace (*argparse.Namespace*) The result from CLI parsing.

Returns
An auth plugin, or None if a name is not provided.

Return type
keystoneclient.auth.BaseAuthPlugin

keystoneclient.auth.identity.generic.password module

class keystoneclient.auth.identity.generic.password.**Password**(*auth_url*, *username=None*, *user_id=None*, *password=None*, *user_domain_id=None*, *user_domain_name=None*, ***kwargs*)

Bases: *BaseGenericPlugin*

A common user/password authentication plugin.

Parameters

- **username** (*string*) Username for authentication.
- **user_id** (*string*) User ID for authentication.
- **password** (*string*) Password for authentication.
- **user_domain_id** (*string*) Users domain ID for authentication.

- **user_domain_name** (*string*) Users domain name for authentication.

create_plugin(*session*, *version*, *url*, *raw_status=None*)

Create a plugin from the given parameters.

This function will be called multiple times with the version and url of a potential endpoint. If a plugin can be constructed that fits the params then it should return it. If not return None and then another call will be made with other available URLs.

Parameters

- **session** (keystoneclient.session.Session) A session object.
- **version** (*tuple*) A tuple of the API version at the URL.
- **url** (*string*) The base URL for this version.
- **raw_status** (*string*) The status that was in the discovery field.

Returns
A plugin that can match the parameters or None if nothing.

classmethod get_options()

Return the list of parameters associated with the auth plugin.

This list may be used to generate CLI or config arguments.

Returns
A list of Param objects describing available plugin parameters.

Return type
List

classmethod load_from_argparse_arguments(*namespace*, ***kwargs*)

Load a specific plugin object from an argparse result.

Convert the results of a parse into the specified plugin.

Parameters
namespace (*argparse.Namespace*) The result from CLI parsing.

Returns
An auth plugin, or None if a name is not provided.

Return type
keystoneclient.auth.BaseAuthPlugin

keystoneclient.auth.identity.generic.password.**get_options**()

keystoneclient.auth.identity.generic.token module

class keystoneclient.auth.identity.generic.token.**Token**(*auth_url*, *token=None*, ***kwargs*)

Bases: *BaseGenericPlugin*

Generic token auth plugin.

Parameters
token (*string*) Token for authentication.

create_plugin(*session*, *version*, *url*, *raw_status=None*)

Create a plugin from the given parameters.

This function will be called multiple times with the version and url of a potential endpoint. If a plugin can be constructed that fits the params then it should return it. If not return None and then another call will be made with other available URLs.

Parameters

- **session** (keystoneclient.session.Session) A session object.
- **version** (*tuple*) A tuple of the API version at the URL.
- **url** (*string*) The base URL for this version.
- **raw_status** (*string*) The status that was in the discovery field.

Returns
A plugin that can match the parameters or None if nothing.

classmethod get_options()

Return the list of parameters associated with the auth plugin.

This list may be used to generate CLI or config arguments.

Returns
A list of Param objects describing available plugin parameters.

Return type
List

keystoneclient.auth.identity.generic.token.**get_options**()

Module contents

class keystoneclient.auth.identity.generic.**BaseGenericPlugin**(*auth_url*, *tenant_id=None*, *tenant_name=None*, *project_id=None*, *project_name=None*, *project_domain_id=None*, *project_domain_name=None*, *domain_id=None*, *domain_name=None*, *trust_id=None*)

Bases: *BaseIdentityPlugin*

An identity plugin that is not version dependent.

Internally we will construct a version dependent plugin with the resolved URL and then proxy all calls from the base plugin to the versioned one.

abstract create_plugin(*session*, *version*, *url*, *raw_status=None*)

Create a plugin from the given parameters.

This function will be called multiple times with the version and url of a potential endpoint. If a plugin can be constructed that fits the params then it should return it. If not return None and then another call will be made with other available URLs.

Parameters

- **session** (keystoneclient.session.Session) A session object.
- **version** (*tuple*) A tuple of the API version at the URL.
- **url** (*string*) The base URL for this version.
- **raw_status** (*string*) The status that was in the discovery field.

Returns
A plugin that can match the parameters or None if nothing.

get_auth_ref(*session*, ***kwargs*)

Obtain a token from an OpenStack Identity Service.

This method is overridden by the various token version plugins.

This method should not be called independently and is expected to be invoked via the do_authenticate() method.

This method will be invoked if the AccessInfo object cached by the plugin is not valid. Thus plugins should always fetch a new AccessInfo when invoked. If you are looking to just retrieve the current auth data then you should use get_access().

Parameters
session (keystoneclient.session.Session) A session object that can be used for communication.

Raises

- *keystoneclient.exceptions.InvalidResponse* The response returned wasnt appropriate.
- *keystoneclient.exceptions.HttpError* An error from an invalid HTTP response.

Returns
Token access information.

Return type
keystoneclient.access.AccessInfo

classmethod get_options()

Return the list of parameters associated with the auth plugin.

This list may be used to generate CLI or config arguments.

Returns
A list of Param objects describing available plugin parameters.

Return type
List

property trust_id

Deprecated as of the 1.7.0 release.

It may be removed in the 2.0.0 release.

class keystoneclient.auth.identity.generic.**Password**(*auth_url*, *username=None*, *user_id=None*, *password=None*, *user_domain_id=None*, *user_domain_name=None*, ***kwargs*)

Bases: *BaseGenericPlugin*

A common user/password authentication plugin.

Parameters

- **username** (*string*) Username for authentication.
- **user_id** (*string*) User ID for authentication.
- **password** (*string*) Password for authentication.
- **user_domain_id** (*string*) Users domain ID for authentication.
- **user_domain_name** (*string*) Users domain name for authentication.

create_plugin(*session*, *version*, *url*, *raw_status=None*)

Create a plugin from the given parameters.

This function will be called multiple times with the version and url of a potential endpoint. If a plugin can be constructed that fits the params then it should return it. If not return None and then another call will be made with other available URLs.

Parameters

- **session** (keystoneclient.session.Session) A session object.
- **version** (*tuple*) A tuple of the API version at the URL.
- **url** (*string*) The base URL for this version.
- **raw_status** (*string*) The status that was in the discovery field.

Returns

A plugin that can match the parameters or None if nothing.

classmethod get_options()

Return the list of parameters associated with the auth plugin.

This list may be used to generate CLI or config arguments.

Returns

A list of Param objects describing available plugin parameters.

Return type

List

classmethod load_from_argparse_arguments(*namespace*, ***kwargs*)

Load a specific plugin object from an argparse result.

Convert the results of a parse into the specified plugin.

Parameters

namespace (*argparse.Namespace*) The result from CLI parsing.

Returns

An auth plugin, or None if a name is not provided.

Return type
keystoneclient.auth.BaseAuthPlugin

class keystoneclient.auth.identity.generic.**Token**(*auth_url*, *token=None*, ***kwargs*)

Bases: *BaseGenericPlugin*

Generic token auth plugin.

Parameters
token (*string*) Token for authentication.

create_plugin(*session*, *version*, *url*, *raw_status=None*)

Create a plugin from the given parameters.

This function will be called multiple times with the version and url of a potential endpoint. If a plugin can be constructed that fits the params then it should return it. If not return None and then another call will be made with other available URLs.

Parameters

- **session** (keystoneclient.session.Session) A session object.
- **version** (*tuple*) A tuple of the API version at the URL.
- **url** (*string*) The base URL for this version.
- **raw_status** (*string*) The status that was in the discovery field.

Returns
A plugin that can match the parameters or None if nothing.

classmethod get_options()

Return the list of parameters associated with the auth plugin.

This list may be used to generate CLI or config arguments.

Returns
A list of Param objects describing available plugin parameters.

Return type
List

keystoneclient.auth.identity.v3 package

Submodules

keystoneclient.auth.identity.v3.base module

class keystoneclient.auth.identity.v3.base.**Auth**(*auth_url*, *auth_methods*, ***kwargs*)

Bases: *BaseAuth*

Identity V3 Authentication Plugin.

Parameters

- **auth_url** (*string*) Identity service endpoint for authentication.
- **auth_methods** (*List*) A collection of methods to authenticate with.

- **trust_id** (*string*) Trust ID for trust scoping.
- **domain_id** (*string*) Domain ID for domain scoping.
- **domain_name** (*string*) Domain name for domain scoping.
- **project_id** (*string*) Project ID for project scoping.
- **project_name** (*string*) Project name for project scoping.
- **project_domain_id** (*string*) Projects domain ID for project.
- **project_domain_name** (*string*) Projects domain name for project.
- **reauthenticate** (*bool*) Allow fetching a new token if the current one is going to expire. (optional) default True
- **include_catalog** (*bool*) Include the service catalog in the returned token. (optional) default True.
- **unscoped** (*bool*) Force the return of an unscoped token. This will make the keystone server return an unscoped token even if a default_project_id is set for this user.

get_auth_ref(*session*, ***kwargs*)

Obtain a token from an OpenStack Identity Service.

This method is overridden by the various token version plugins.

This method should not be called independently and is expected to be invoked via the do_authenticate() method.

This method will be invoked if the AccessInfo object cached by the plugin is not valid. Thus plugins should always fetch a new AccessInfo when invoked. If you are looking to just retrieve the current auth data then you should use get_access().

Parameters

session (keystoneclient.session.Session) A session object that can be used for communication.

Raises

- *keystoneclient.exceptions.InvalidResponse* The response returned wasnt appropriate.
- *keystoneclient.exceptions.HttpError* An error from an invalid HTTP response.

Returns

Token access information.

Return type

keystoneclient.access.AccessInfo

class keystoneclient.auth.identity.v3.base.**AuthConstructor**(*auth_url*, **args*, ***kwargs*)

Bases: *Auth*

Abstract base class for creating an Auth Plugin.

The Auth Plugin created contains only one authentication method. This is generally the required usage.

An AuthConstructor creates an AuthMethod based on the methods arguments and the auth_method_class defined by the plugin. It then creates the auth plugin with only that authentication method.

class keystoneclient.auth.identity.v3.base.**AuthMethod**(***kwargs*)

Bases: object

One part of a V3 Authentication strategy.

V3 Tokens allow multiple methods to be presented when authentication against the server. Each one of these methods is implemented by an AuthMethod.

Note: When implementing an AuthMethod use the method_parameters and do not use positional arguments. Otherwise they cant be picked up by the factory method and dont work as well with AuthConstructors.

abstract get_auth_data(*session*, *auth*, *headers*, ***kwargs*)

Return the authentication section of an auth plugin.

Parameters

- **session** (keystoneclient.session.Session) The communication session.
- **auth** (base.Auth) The auth plugin calling the method.
- **headers** (*dict*) The headers that will be sent with the auth request if a plugin needs to add to them.

Returns

The identifier of this plugin and a dict of authentication data for the auth type.

Return type

tuple(string, dict)

class keystoneclient.auth.identity.v3.base.**BaseAuth**(*auth_url*, *trust_id=None*, *domain_id=None*, *domain_name=None*, *project_id=None*, *project_name=None*, *project_domain_id=None*, *project_domain_name=None*, *reauthenticate=True*, *include_catalog=True*)

Bases: *BaseIdentityPlugin*

Identity V3 Authentication Plugin.

Parameters

- **auth_url** (*string*) Identity service endpoint for authentication.
- **auth_methods** (*List*) A collection of methods to authenticate with.
- **trust_id** (*string*) Trust ID for trust scoping.
- **domain_id** (*string*) Domain ID for domain scoping.
- **domain_name** (*string*) Domain name for domain scoping.
- **project_id** (*string*) Project ID for project scoping.

- **project_name** (*string*) Project name for project scoping.
- **project_domain_id** (*string*) Projects domain ID for project.
- **project_domain_name** (*string*) Projects domain name for project.
- **reauthenticate** (*bool*) Allow fetching a new token if the current one is going to expire. (optional) default True
- **include_catalog** (*bool*) Include the service catalog in the returned token. (optional) default True.

abstract get_auth_ref(*session*, ***kwargs*)

Obtain a token from an OpenStack Identity Service.

This method is overridden by the various token version plugins.

This method should not be called independently and is expected to be invoked via the do_authenticate() method.

This method will be invoked if the AccessInfo object cached by the plugin is not valid. Thus plugins should always fetch a new AccessInfo when invoked. If you are looking to just retrieve the current auth data then you should use get_access().

Parameters

session (keystoneclient.session.Session) A session object that can be used for communication.

Raises

- *keystoneclient.exceptions.InvalidResponse* The response returned wasnt appropriate.
- *keystoneclient.exceptions.HttpError* An error from an invalid HTTP response.

Returns

Token access information.

Return type

keystoneclient.access.AccessInfo

classmethod get_options()

Return the list of parameters associated with the auth plugin.

This list may be used to generate CLI or config arguments.

Returns

A list of Param objects describing available plugin parameters.

Return type

List

property token_url

The full URL where we will send authentication data.

property trust_id

Deprecated as of the 1.7.0 release.

It may be removed in the 2.0.0 release.

keystoneclient.auth.identity.v3.federated module

class keystoneclient.auth.identity.v3.federated.**FederatedBaseAuth**(*auth_url*, *identity_provider*, *protocol*, ***kwargs*)

Bases: *BaseAuth*

property federated_token_url

Full URL where authorization data is sent.

get_auth_ref(*session*, ***kwargs*)

Authenticate retrieve token information.

This is a multi-step process where a client does federated authn receives an unscoped token.

If an unscoped token is successfully received and scoping information is present then the token is rescoped to that target.

Parameters
session (keystoneclient.session.Session) a session object to send out HTTP requests.

Returns
a token data representation

Return type
keystoneclient.access.AccessInfo

classmethod get_options()

Return the list of parameters associated with the auth plugin.

This list may be used to generate CLI or config arguments.

Returns
A list of Param objects describing available plugin parameters.

Return type
List

abstract get_unscoped_auth_ref(*session*, ***kwargs*)

Fetch unscoped federated token.

rescoping_plugin

alias of *Token*

keystoneclient.auth.identity.v3.password module

class keystoneclient.auth.identity.v3.password.**Password**(*auth_url*, **args*, ***kwargs*)

Bases: *AuthConstructor*

A plugin for authenticating with a username and password.

Parameters

- **auth_url** (*string*) Identity service endpoint for authentication.

- **password** (*string*) Password for authentication.
- **username** (*string*) Username for authentication.
- **user_id** (*string*) User ID for authentication.
- **user_domain_id** (*string*) Users domain ID for authentication.
- **user_domain_name** (*string*) Users domain name for authentication.
- **trust_id** (*string*) Trust ID for trust scoping.
- **domain_id** (*string*) Domain ID for domain scoping.
- **domain_name** (*string*) Domain name for domain scoping.
- **project_id** (*string*) Project ID for project scoping.
- **project_name** (*string*) Project name for project scoping.
- **project_domain_id** (*string*) Projects domain ID for project.
- **project_domain_name** (*string*) Projects domain name for project.
- **reauthenticate** (*bool*) Allow fetching a new token if the current one is going to expire. (optional) default True

classmethod get_options()

Return the list of parameters associated with the auth plugin.

This list may be used to generate CLI or config arguments.

Returns
A list of Param objects describing available plugin parameters.

Return type
List

classmethod load_from_argparse_arguments(*namespace*, ***kwargs*)

Load a specific plugin object from an argparse result.

Convert the results of a parse into the specified plugin.

Parameters
namespace (*argparse.Namespace*) The result from CLI parsing.

Returns
An auth plugin, or None if a name is not provided.

Return type
keystoneclient.auth.BaseAuthPlugin

class keystoneclient.auth.identity.v3.password.**PasswordMethod**(***kwargs*)

Bases: *AuthMethod*

Construct a User/Password based authentication method.

Parameters

- **password** (*string*) Password for authentication.
- **username** (*string*) Username for authentication.
- **user_id** (*string*) User ID for authentication.

- **user_domain_id** (*string*) Users domain ID for authentication.
- **user_domain_name** (*string*) Users domain name for authentication.

get_auth_data(*session*, *auth*, *headers*, ***kwargs*)

Return the authentication section of an auth plugin.

Parameters

- **session** (keystoneclient.session.Session) The communication session.
- **auth** (base.Auth) The auth plugin calling the method.
- **headers** (*dict*) The headers that will be sent with the auth request if a plugin needs to add to them.

Returns

The identifier of this plugin and a dict of authentication data for the auth type.

Return type

tuple(string, dict)

keystoneclient.auth.identity.v3.token module

class keystoneclient.auth.identity.v3.token.**Token**(*auth_url*, *token*, ***kwargs*)

Bases: *AuthConstructor*

A plugin for authenticating with an existing Token.

Parameters

- **auth_url** (*string*) Identity service endpoint for authentication.
- **token** (*string*) Token for authentication.
- **trust_id** (*string*) Trust ID for trust scoping.
- **domain_id** (*string*) Domain ID for domain scoping.
- **domain_name** (*string*) Domain name for domain scoping.
- **project_id** (*string*) Project ID for project scoping.
- **project_name** (*string*) Project name for project scoping.
- **project_domain_id** (*string*) Projects domain ID for project.
- **project_domain_name** (*string*) Projects domain name for project.
- **reauthenticate** (*bool*) Allow fetching a new token if the current one is going to expire. (optional) default True

classmethod get_options()

Return the list of parameters associated with the auth plugin.

This list may be used to generate CLI or config arguments.

Returns

A list of Param objects describing available plugin parameters.

Return type
List

class keystoneclient.auth.identity.v3.token.**TokenMethod**(***kwargs*)

Bases: *AuthMethod*

Construct an Auth plugin to fetch a token from a token.

Parameters
token (*string*) Token for authentication.

get_auth_data(*session*, *auth*, *headers*, ***kwargs*)

Return the authentication section of an auth plugin.

Parameters

- **session** (keystoneclient.session.Session) The communication session.
- **auth** (base.Auth) The auth plugin calling the method.
- **headers** (*dict*) The headers that will be sent with the auth request if a plugin needs to add to them.

Returns
The identifier of this plugin and a dict of authentication data for the auth type.

Return type
tuple(string, dict)

Module contents

class keystoneclient.auth.identity.v3.**Auth**(*auth_url*, *auth_methods*, ***kwargs*)

Bases: *BaseAuth*

Identity V3 Authentication Plugin.

Parameters

- **auth_url** (*string*) Identity service endpoint for authentication.
- **auth_methods** (*List*) A collection of methods to authenticate with.
- **trust_id** (*string*) Trust ID for trust scoping.
- **domain_id** (*string*) Domain ID for domain scoping.
- **domain_name** (*string*) Domain name for domain scoping.
- **project_id** (*string*) Project ID for project scoping.
- **project_name** (*string*) Project name for project scoping.
- **project_domain_id** (*string*) Projects domain ID for project.
- **project_domain_name** (*string*) Projects domain name for project.
- **reauthenticate** (*bool*) Allow fetching a new token if the current one is going to expire. (optional) default True

- **include_catalog** (*bool*) Include the service catalog in the returned token. (optional) default True.
- **unscoped** (*bool*) Force the return of an unscoped token. This will make the keystone server return an unscoped token even if a default_project_id is set for this user.

get_auth_ref(*session*, ***kwargs*)

Obtain a token from an OpenStack Identity Service.

This method is overridden by the various token version plugins.

This method should not be called independently and is expected to be invoked via the do_authenticate() method.

This method will be invoked if the AccessInfo object cached by the plugin is not valid. Thus plugins should always fetch a new AccessInfo when invoked. If you are looking to just retrieve the current auth data then you should use get_access().

Parameters

session (keystoneclient.session.Session) A session object that can be used for communication.

Raises

- *keystoneclient.exceptions.InvalidResponse* The response returned wasnt appropriate.
- *keystoneclient.exceptions.HttpError* An error from an invalid HTTP response.

Returns

Token access information.

Return type

keystoneclient.access.AccessInfo

class keystoneclient.auth.identity.v3.**AuthConstructor**(*auth_url*, **args*, ***kwargs*)

Bases: *Auth*

Abstract base class for creating an Auth Plugin.

The Auth Plugin created contains only one authentication method. This is generally the required usage.

An AuthConstructor creates an AuthMethod based on the methods arguments and the auth_method_class defined by the plugin. It then creates the auth plugin with only that authentication method.

class keystoneclient.auth.identity.v3.**AuthMethod**(***kwargs*)

Bases: object

One part of a V3 Authentication strategy.

V3 Tokens allow multiple methods to be presented when authentication against the server. Each one of these methods is implemented by an AuthMethod.

Note: When implementing an AuthMethod use the method_parameters and do not use positional arguments. Otherwise they cant be picked up by the factory method and dont work as well with AuthConstructors.

abstract get_auth_data(*session*, *auth*, *headers*, ***kwargs*)

Return the authentication section of an auth plugin.

Parameters

- **session** (keystoneclient.session.Session) The communication session.
- **auth** (base.Auth) The auth plugin calling the method.
- **headers** (*dict*) The headers that will be sent with the auth request if a plugin needs to add to them.

Returns

The identifier of this plugin and a dict of authentication data for the auth type.

Return type

tuple(string, dict)

class keystoneclient.auth.identity.v3.**BaseAuth**(*auth_url*, *trust_id=None*, *domain_id=None*, *domain_name=None*, *project_id=None*, *project_name=None*, *project_domain_id=None*, *project_domain_name=None*, *reauthenticate=True*, *include_catalog=True*)

Bases: *BaseIdentityPlugin*

Identity V3 Authentication Plugin.

Parameters

- **auth_url** (*string*) Identity service endpoint for authentication.
- **auth_methods** (*List*) A collection of methods to authenticate with.
- **trust_id** (*string*) Trust ID for trust scoping.
- **domain_id** (*string*) Domain ID for domain scoping.
- **domain_name** (*string*) Domain name for domain scoping.
- **project_id** (*string*) Project ID for project scoping.
- **project_name** (*string*) Project name for project scoping.
- **project_domain_id** (*string*) Projects domain ID for project.
- **project_domain_name** (*string*) Projects domain name for project.
- **reauthenticate** (*bool*) Allow fetching a new token if the current one is going to expire. (optional) default True
- **include_catalog** (*bool*) Include the service catalog in the returned token. (optional) default True.

abstract get_auth_ref(*session*, ***kwargs*)

Obtain a token from an OpenStack Identity Service.

This method is overridden by the various token version plugins.

This method should not be called independently and is expected to be invoked via the do_authenticate() method.

This method will be invoked if the AccessInfo object cached by the plugin is not valid. Thus plugins should always fetch a new AccessInfo when invoked. If you are looking to just retrieve the current auth data then you should use get_access().

Parameters
session (keystoneclient.session.Session) A session object that can be used for communication.

Raises
- *keystoneclient.exceptions.InvalidResponse* The response returned wasnt appropriate.
- *keystoneclient.exceptions.HttpError* An error from an invalid HTTP response.

Returns
Token access information.

Return type
keystoneclient.access.AccessInfo

classmethod get_options()

Return the list of parameters associated with the auth plugin.

This list may be used to generate CLI or config arguments.

Returns
A list of Param objects describing available plugin parameters.

Return type
List

property token_url

The full URL where we will send authentication data.

property trust_id

Deprecated as of the 1.7.0 release.

It may be removed in the 2.0.0 release.

class keystoneclient.auth.identity.v3.FederatedBaseAuth(*auth_url*, *identity_provider*, *protocol*, ***kwargs*)

Bases: *BaseAuth*

property federated_token_url

Full URL where authorization data is sent.

get_auth_ref(*session*, ***kwargs*)

Authenticate retrieve token information.

This is a multi-step process where a client does federated authn receives an unscoped token.

If an unscoped token is successfully received and scoping information is present then the token is rescoped to that target.

Parameters
session (keystoneclient.session.Session) a session object to send out HTTP requests.

Returns
a token data representation

Return type
keystoneclient.access.AccessInfo

classmethod get_options()

Return the list of parameters associated with the auth plugin.

This list may be used to generate CLI or config arguments.

Returns
A list of Param objects describing available plugin parameters.

Return type
List

abstract get_unscoped_auth_ref(*session*, ***kwargs*)

Fetch unscoped federated token.

rescoping_plugin

alias of *Token*

class keystoneclient.auth.identity.v3.**Password**(*auth_url*, **args*, ***kwargs*)

Bases: *AuthConstructor*

A plugin for authenticating with a username and password.

Parameters

- **auth_url** (*string*) Identity service endpoint for authentication.
- **password** (*string*) Password for authentication.
- **username** (*string*) Username for authentication.
- **user_id** (*string*) User ID for authentication.
- **user_domain_id** (*string*) Users domain ID for authentication.
- **user_domain_name** (*string*) Users domain name for authentication.
- **trust_id** (*string*) Trust ID for trust scoping.
- **domain_id** (*string*) Domain ID for domain scoping.
- **domain_name** (*string*) Domain name for domain scoping.
- **project_id** (*string*) Project ID for project scoping.
- **project_name** (*string*) Project name for project scoping.
- **project_domain_id** (*string*) Projects domain ID for project.
- **project_domain_name** (*string*) Projects domain name for project.
- **reauthenticate** (*bool*) Allow fetching a new token if the current one is going to expire. (optional) default True

classmethod get_options()

Return the list of parameters associated with the auth plugin.

This list may be used to generate CLI or config arguments.

Returns
A list of Param objects describing available plugin parameters.

Return type
List

classmethod load_from_argparse_arguments(*namespace*, ***kwargs*)

Load a specific plugin object from an argparse result.

Convert the results of a parse into the specified plugin.

Parameters
namespace (*argparse.Namespace*) The result from CLI parsing.

Returns
An auth plugin, or None if a name is not provided.

Return type
keystoneclient.auth.BaseAuthPlugin

class keystoneclient.auth.identity.v3.**PasswordMethod**(***kwargs*)

Bases: *AuthMethod*

Construct a User/Password based authentication method.

Parameters

- **password** (*string*) Password for authentication.
- **username** (*string*) Username for authentication.
- **user_id** (*string*) User ID for authentication.
- **user_domain_id** (*string*) Users domain ID for authentication.
- **user_domain_name** (*string*) Users domain name for authentication.

get_auth_data(*session*, *auth*, *headers*, ***kwargs*)

Return the authentication section of an auth plugin.

Parameters

- **session** (keystoneclient.session.Session) The communication session.
- **auth** (base.Auth) The auth plugin calling the method.
- **headers** (*dict*) The headers that will be sent with the auth request if a plugin needs to add to them.

Returns
The identifier of this plugin and a dict of authentication data for the auth type.

Return type
tuple(string, dict)

class keystoneclient.auth.identity.v3.**Token**(*auth_url*, *token*, ***kwargs*)

Bases: *AuthConstructor*

A plugin for authenticating with an existing Token.

Parameters

- **auth_url** (*string*) Identity service endpoint for authentication.
- **token** (*string*) Token for authentication.
- **trust_id** (*string*) Trust ID for trust scoping.
- **domain_id** (*string*) Domain ID for domain scoping.
- **domain_name** (*string*) Domain name for domain scoping.
- **project_id** (*string*) Project ID for project scoping.
- **project_name** (*string*) Project name for project scoping.
- **project_domain_id** (*string*) Projects domain ID for project.
- **project_domain_name** (*string*) Projects domain name for project.
- **reauthenticate** (*bool*) Allow fetching a new token if the current one is going to expire. (optional) default True

classmethod get_options()

Return the list of parameters associated with the auth plugin.

This list may be used to generate CLI or config arguments.

Returns
A list of Param objects describing available plugin parameters.

Return type
List

class keystoneclient.auth.identity.v3.**TokenMethod**(***kwargs*)

Bases: *AuthMethod*

Construct an Auth plugin to fetch a token from a token.

Parameters
token (*string*) Token for authentication.

get_auth_data(*session*, *auth*, *headers*, ***kwargs*)

Return the authentication section of an auth plugin.

Parameters

- **session** (keystoneclient.session.Session) The communication session.
- **auth** (base.Auth) The auth plugin calling the method.
- **headers** (*dict*) The headers that will be sent with the auth request if a plugin needs to add to them.

Returns
The identifier of this plugin and a dict of authentication data for the auth type.

Return type
tuple(string, dict)

Submodules

keystoneclient.auth.identity.access module

class keystoneclient.auth.identity.access.**AccessInfoPlugin**(*auth_ref*, *auth_url=None*)

Bases: *BaseIdentityPlugin*

A plugin that turns an existing AccessInfo object into a usable plugin.

There are cases where reuse of an auth_ref or AccessInfo object is warranted such as from a cache, from auth_token middleware, or another source.

Turn the existing access info object into an identity plugin. This plugin cannot be refreshed as the AccessInfo object does not contain any authorizing information.

Parameters

- **auth_ref** (keystoneclient.access.AccessInfo) the existing AccessInfo object.
- **auth_url** the url where this AccessInfo was retrieved from. Required if using the AUTH_INTERFACE with get_endpoint. (optional)

get_auth_ref(*session*, ***kwargs*)

Obtain a token from an OpenStack Identity Service.

This method is overridden by the various token version plugins.

This method should not be called independently and is expected to be invoked via the do_authenticate() method.

This method will be invoked if the AccessInfo object cached by the plugin is not valid. Thus plugins should always fetch a new AccessInfo when invoked. If you are looking to just retrieve the current auth data then you should use get_access().

Parameters
session (keystoneclient.session.Session) A session object that can be used for communication.

Raises

- *keystoneclient.exceptions.InvalidResponse* The response returned wasnt appropriate.
- *keystoneclient.exceptions.HttpError* An error from an invalid HTTP response.

Returns
Token access information.

Return type
keystoneclient.access.AccessInfo

invalidate()

Invalidate the current authentication data.

This should result in fetching a new token on next call.

A plugin may be invalidated if an Unauthorized HTTP response is returned to indicate that the token may have been revoked or is otherwise now invalid.

Returns
True if there was something that the plugin did to invalidate. This means that it makes sense to try again. If nothing happens returns False to indicate give up.

Return type
bool

keystoneclient.auth.identity.base module

class keystoneclient.auth.identity.base.**BaseIdentityPlugin**(*auth_url=None*, *username=None*, *password=None*, *token=None*, *trust_id=None*, *reauthenticate=True*)

Bases: *BaseAuthPlugin*

MIN_TOKEN_LIFE_SECONDS = 120

get_access(*session*, ***kwargs*)

Fetch or return a current AccessInfo object.

If a valid AccessInfo is present then it is returned otherwise a new one will be fetched.

Parameters
session (keystoneclient.session.Session) A session object that can be used for communication.

Raises
keystoneclient.exceptions.HttpError An error from an invalid HTTP response.

Returns
Valid AccessInfo

Return type
keystoneclient.access.AccessInfo

abstract get_auth_ref(*session*, ***kwargs*)

Obtain a token from an OpenStack Identity Service.

This method is overridden by the various token version plugins.

This method should not be called independently and is expected to be invoked via the do_authenticate() method.

This method will be invoked if the AccessInfo object cached by the plugin is not valid. Thus plugins should always fetch a new AccessInfo when invoked. If you are looking to just retrieve the current auth data then you should use get_access().

Parameters
session (keystoneclient.session.Session) A session object that can be used for communication.

Raises
- *keystoneclient.exceptions.InvalidResponse* The response returned wasnt appropriate.
- *keystoneclient.exceptions.HttpError* An error from an invalid HTTP response.

Returns
Token access information.

Return type
keystoneclient.access.AccessInfo

get_discovery(*session*, *url*, *authenticated=None*)

Return the discovery object for a URL.

Check the session and the plugin cache to see if we have already performed discovery on the URL and if so return it, otherwise create a new discovery object, cache it and return it.

This function is expected to be used by subclasses and should not be needed by users.

Parameters
- **session** (keystoneclient.session.Session) A session object to discover with.
- **url** (*str*) The url to lookup.
- **authenticated** (*bool*) Include a token in the discovery call. (optional) Defaults to None (use a token if a plugin is installed).

Raises
- *keystoneclient.exceptions.DiscoveryFailure* if for some reason the lookup fails.
- *keystoneclient.exceptions.HttpError* An error from an invalid HTTP response.

Returns
A discovery object with the results of looking up that URL.

get_endpoint(*session*, *service_type=None*, *interface=None*, *region_name=None*, *service_name=None*, *version=None*, ***kwargs*)

Return a valid endpoint for a service.

If a valid token is not present then a new one will be fetched using the session and kwargs.

Parameters
- **session** (keystoneclient.session.Session) A session object that can be used for communication.
- **service_type** (*string*) The type of service to lookup the endpoint for. This plugin will return None (failure) if service_type is not provided.

- **interface** (*string*) The exposure of the endpoint. Should be *public*, *internal*, *admin*, or *auth*. *auth* is special here to use the *auth_url* rather than a URL extracted from the service catalog. Defaults to *public*.
- **region_name** (*string*) The region the endpoint should exist in. (optional)
- **service_name** (*string*) The name of the service in the catalog. (optional)
- **version** (*tuple*) The minimum version number required for this endpoint. (optional)

Raises
keystoneclient.exceptions.HttpError An error from an invalid HTTP response.

Returns
A valid endpoint URL or None if not available.

Return type
string or None

classmethod get_options()

Return the list of parameters associated with the auth plugin.

This list may be used to generate CLI or config arguments.

Returns
A list of Param objects describing available plugin parameters.

Return type
List

get_project_id(*session*, ***kwargs*)

Return the project id that we are authenticated to.

Wherever possible the project id should be inferred from the token however there are certain URLs and other places that require access to the currently authenticated project id.

Parameters
session (keystoneclient.session.Session) A session object so the plugin can make HTTP calls.

Returns
A project identifier or None if one is not available.

Return type
str

get_token(*session*, ***kwargs*)

Return a valid auth token.

If a valid token is not present then a new one will be fetched.

Parameters
session (keystoneclient.session.Session) A session object that can be used for communication.

Raises
keystoneclient.exceptions.HttpError An error from an invalid HTTP response.

Returns
A valid token.

Return type
string

`get_user_id`(*session*, ***kwargs*)

Return a unique user identifier of the plugin.

Wherever possible the user id should be inferred from the token however there are certain URLs and other places that require access to the currently authenticated user id.

Parameters
session (keystoneclient.session.Session) A session object so the plugin can make HTTP calls.

Returns
A user identifier or None if one is not available.

Return type
str

`invalidate`()

Invalidate the current authentication data.

This should result in fetching a new token on next call.

A plugin may be invalidated if an Unauthorized HTTP response is returned to indicate that the token may have been revoked or is otherwise now invalid.

Returns
True if there was something that the plugin did to invalidate. This means that it makes sense to try again. If nothing happens returns False to indicate give up.

Return type
bool

`property password`

Deprecated as of the 1.7.0 release.

It may be removed in the 2.0.0 release.

`property token`

Deprecated as of the 1.7.0 release.

It may be removed in the 2.0.0 release.

`property trust_id`

Deprecated as of the 1.7.0 release.

It may be removed in the 2.0.0 release.

`property username`

Deprecated as of the 1.7.0 release.

It may be removed in the 2.0.0 release.

keystoneclient.auth.identity.base.`get_options`()

keystoneclient.auth.identity.v2 module

class keystoneclient.auth.identity.v2.**Auth**(*auth_url*, *trust_id=None*, *tenant_id=None*, *tenant_name=None*, *reauthenticate=True*)

Bases: *BaseIdentityPlugin*

Identity V2 Authentication Plugin.

Parameters

- **auth_url** (*string*) Identity service endpoint for authorization.
- **trust_id** (*string*) Trust ID for trust scoping.
- **tenant_id** (*string*) Tenant ID for project scoping.
- **tenant_name** (*string*) Tenant name for project scoping.
- **reauthenticate** (*bool*) Allow fetching a new token if the current one is going to expire. (optional) default True

abstract get_auth_data(*headers=None*)

Return the authentication section of an auth plugin.

Parameters

headers (*dict*) The headers that will be sent with the auth request if a plugin needs to add to them.

Returns

A dict of authentication data for the auth type.

Return type

dict

get_auth_ref(*session*, ***kwargs*)

Obtain a token from an OpenStack Identity Service.

This method is overridden by the various token version plugins.

This method should not be called independently and is expected to be invoked via the do_authenticate() method.

This method will be invoked if the AccessInfo object cached by the plugin is not valid. Thus plugins should always fetch a new AccessInfo when invoked. If you are looking to just retrieve the current auth data then you should use get_access().

Parameters

session (*keystoneclient.session.Session*) A session object that can be used for communication.

Raises

- *keystoneclient.exceptions.InvalidResponse* The response returned wasnt appropriate.
- *keystoneclient.exceptions.HttpError* An error from an invalid HTTP response.

Returns

Token access information.

Return type
keystoneclient.access.AccessInfo

classmethod get_options()

Return the list of parameters associated with the auth plugin.

This list may be used to generate CLI or config arguments.

Returns
A list of Param objects describing available plugin parameters.

Return type
List

property trust_id

Deprecated as of the 1.7.0 release.

It may be removed in the 2.0.0 release.

class keystoneclient.auth.identity.v2.Password(*auth_url*, *username=<object object>*, *password=None*, *user_id=<object object>*, ***kwargs*)

Bases: *Auth*

A plugin for authenticating with a username and password.

A username or user_id must be provided.

Parameters

- **auth_url** (*string*) Identity service endpoint for authorization.
- **username** (*string*) Username for authentication.
- **password** (*string*) Password for authentication.
- **user_id** (*string*) User ID for authentication.
- **trust_id** (*string*) Trust ID for trust scoping.
- **tenant_id** (*string*) Tenant ID for tenant scoping.
- **tenant_name** (*string*) Tenant name for tenant scoping.
- **reauthenticate** (*bool*) Allow fetching a new token if the current one is going to expire. (optional) default True

Raises
TypeError if a user_id or username is not provided.

get_auth_data(*headers=None*)

Return the authentication section of an auth plugin.

Parameters
headers (*dict*) The headers that will be sent with the auth request if a plugin needs to add to them.

Returns
A dict of authentication data for the auth type.

Return type
dict

classmethod get_options()

Return the list of parameters associated with the auth plugin.

This list may be used to generate CLI or config arguments.

Returns
A list of Param objects describing available plugin parameters.

Return type
List

classmethod load_from_argparse_arguments(*namespace*, ***kwargs*)

Load a specific plugin object from an argparse result.

Convert the results of a parse into the specified plugin.

Parameters
namespace (*argparse.Namespace*) The result from CLI parsing.

Returns
An auth plugin, or None if a name is not provided.

Return type
keystoneclient.auth.BaseAuthPlugin

property password

Deprecated as of the 1.7.0 release.

It may be removed in the 2.0.0 release.

property username

Deprecated as of the 1.7.0 release.

It may be removed in the 2.0.0 release.

class keystoneclient.auth.identity.v2.**Token**(*auth_url*, *token*, ***kwargs*)

Bases: *Auth*

A plugin for authenticating with an existing token.

Parameters

- **auth_url** (*string*) Identity service endpoint for authorization.
- **token** (*string*) Existing token for authentication.
- **tenant_id** (*string*) Tenant ID for tenant scoping.
- **tenant_name** (*string*) Tenant name for tenant scoping.
- **trust_id** (*string*) Trust ID for trust scoping.
- **reauthenticate** (*bool*) Allow fetching a new token if the current one is going to expire. (optional) default True

get_auth_data(*headers=None*)

Return the authentication section of an auth plugin.

Parameters
headers (*dict*) The headers that will be sent with the auth request if a plugin needs to add to them.

Returns
A dict of authentication data for the auth type.

Return type
dict

classmethod get_options()

Return the list of parameters associated with the auth plugin.

This list may be used to generate CLI or config arguments.

Returns
A list of Param objects describing available plugin parameters.

Return type
List

property token

Deprecated as of the 1.7.0 release.

It may be removed in the 2.0.0 release.

Module contents

class keystoneclient.auth.identity.**BaseIdentityPlugin**(*auth_url=None*, *username=None*, *password=None*, *token=None*, *trust_id=None*, *reauthenticate=True*)

Bases: *BaseAuthPlugin*

MIN_TOKEN_LIFE_SECONDS = 120

get_access(*session*, ***kwargs*)

Fetch or return a current AccessInfo object.

If a valid AccessInfo is present then it is returned otherwise a new one will be fetched.

Parameters
session (keystoneclient.session.Session) A session object that can be used for communication.

Raises
keystoneclient.exceptions.HttpError An error from an invalid HTTP response.

Returns
Valid AccessInfo

Return type
keystoneclient.access.AccessInfo

abstract get_auth_ref(*session*, ***kwargs*)

Obtain a token from an OpenStack Identity Service.

This method is overridden by the various token version plugins.

This method should not be called independently and is expected to be invoked via the do_authenticate() method.

This method will be invoked if the AccessInfo object cached by the plugin is not valid. Thus plugins should always fetch a new AccessInfo when invoked. If you are looking to just retrieve the current auth data then you should use get_access().

Parameters

session (keystoneclient.session.Session) A session object that can be used for communication.

Raises

- *keystoneclient.exceptions.InvalidResponse* The response returned wasnt appropriate.
- *keystoneclient.exceptions.HttpError* An error from an invalid HTTP response.

Returns

Token access information.

Return type

keystoneclient.access.AccessInfo

get_discovery(*session*, *url*, *authenticated=None*)

Return the discovery object for a URL.

Check the session and the plugin cache to see if we have already performed discovery on the URL and if so return it, otherwise create a new discovery object, cache it and return it.

This function is expected to be used by subclasses and should not be needed by users.

Parameters

- **session** (keystoneclient.session.Session) A session object to discover with.
- **url** (*str*) The url to lookup.
- **authenticated** (*bool*) Include a token in the discovery call. (optional) Defaults to None (use a token if a plugin is installed).

Raises

- *keystoneclient.exceptions.DiscoveryFailure* if for some reason the lookup fails.
- *keystoneclient.exceptions.HttpError* An error from an invalid HTTP response.

Returns

A discovery object with the results of looking up that URL.

get_endpoint(*session*, *service_type=None*, *interface=None*, *region_name=None*, *service_name=None*, *version=None*, ***kwargs*)

Return a valid endpoint for a service.

If a valid token is not present then a new one will be fetched using the session and kwargs.

Parameters

- **session** (keystoneclient.session.Session) A session object that can be used for communication.
- **service_type** (*string*) The type of service to lookup the endpoint for. This plugin will return None (failure) if service_type is not provided.
- **interface** (*string*) The exposure of the endpoint. Should be *public*, *internal*, *admin*, or *auth*. *auth* is special here to use the *auth_url* rather than a URL extracted from the service catalog. Defaults to *public*.
- **region_name** (*string*) The region the endpoint should exist in. (optional)
- **service_name** (*string*) The name of the service in the catalog. (optional)
- **version** (*tuple*) The minimum version number required for this endpoint. (optional)

Raises
keystoneclient.exceptions.HttpError An error from an invalid HTTP response.

Returns
A valid endpoint URL or None if not available.

Return type
string or None

classmethod get_options()

Return the list of parameters associated with the auth plugin.

This list may be used to generate CLI or config arguments.

Returns
A list of Param objects describing available plugin parameters.

Return type
List

get_project_id(*session*, ***kwargs*)

Return the project id that we are authenticated to.

Wherever possible the project id should be inferred from the token however there are certain URLs and other places that require access to the currently authenticated project id.

Parameters
session (keystoneclient.session.Session) A session object so the plugin can make HTTP calls.

Returns
A project identifier or None if one is not available.

Return type
str

get_token(*session*, ***kwargs*)

Return a valid auth token.

If a valid token is not present then a new one will be fetched.

Parameters
session (keystoneclient.session.Session) A session object that can be used for communication.

Raises
keystoneclient.exceptions.HttpError An error from an invalid HTTP response.

Returns
A valid token.

Return type
string

`get_user_id`(*session*, ***kwargs*)

Return a unique user identifier of the plugin.

Wherever possible the user id should be inferred from the token however there are certain URLs and other places that require access to the currently authenticated user id.

Parameters
session (keystoneclient.session.Session) A session object so the plugin can make HTTP calls.

Returns
A user identifier or None if one is not available.

Return type
str

`invalidate`()

Invalidate the current authentication data.

This should result in fetching a new token on next call.

A plugin may be invalidated if an Unauthorized HTTP response is returned to indicate that the token may have been revoked or is otherwise now invalid.

Returns
True if there was something that the plugin did to invalidate. This means that it makes sense to try again. If nothing happens returns False to indicate give up.

Return type
bool

`property password`

Deprecated as of the 1.7.0 release.

It may be removed in the 2.0.0 release.

`property token`

Deprecated as of the 1.7.0 release.

It may be removed in the 2.0.0 release.

`property trust_id`

Deprecated as of the 1.7.0 release.

It may be removed in the 2.0.0 release.

property username

Deprecated as of the 1.7.0 release.

It may be removed in the 2.0.0 release.

class keystoneclient.auth.identity.**Password**(*auth_url*, *username=None*, *user_id=None*, *password=None*, *user_domain_id=None*, *user_domain_name=None*, ***kwargs*)

Bases: *BaseGenericPlugin*

A common user/password authentication plugin.

Parameters

- **username** (*string*) Username for authentication.
- **user_id** (*string*) User ID for authentication.
- **password** (*string*) Password for authentication.
- **user_domain_id** (*string*) Users domain ID for authentication.
- **user_domain_name** (*string*) Users domain name for authentication.

create_plugin(*session*, *version*, *url*, *raw_status=None*)

Create a plugin from the given parameters.

This function will be called multiple times with the version and url of a potential endpoint. If a plugin can be constructed that fits the params then it should return it. If not return None and then another call will be made with other available URLs.

Parameters

- **session** (keystoneclient.session.Session) A session object.
- **version** (*tuple*) A tuple of the API version at the URL.
- **url** (*string*) The base URL for this version.
- **raw_status** (*string*) The status that was in the discovery field.

Returns

A plugin that can match the parameters or None if nothing.

classmethod get_options()

Return the list of parameters associated with the auth plugin.

This list may be used to generate CLI or config arguments.

Returns

A list of Param objects describing available plugin parameters.

Return type

List

classmethod load_from_argparse_arguments(*namespace*, ***kwargs*)

Load a specific plugin object from an argparse result.

Convert the results of a parse into the specified plugin.

Parameters

namespace (*argparse.Namespace*) The result from CLI parsing.

Returns
An auth plugin, or None if a name is not provided.

Return type
keystoneclient.auth.BaseAuthPlugin

class keystoneclient.auth.identity.**Token**(*auth_url*, *token=None*, ***kwargs*)

Bases: *BaseGenericPlugin*

Generic token auth plugin.

Parameters
token (*string*) Token for authentication.

create_plugin(*session*, *version*, *url*, *raw_status=None*)

Create a plugin from the given parameters.

This function will be called multiple times with the version and url of a potential endpoint. If a plugin can be constructed that fits the params then it should return it. If not return None and then another call will be made with other available URLs.

Parameters

- **session** (keystoneclient.session.Session) A session object.
- **version** (*tuple*) A tuple of the API version at the URL.
- **url** (*string*) The base URL for this version.
- **raw_status** (*string*) The status that was in the discovery field.

Returns
A plugin that can match the parameters or None if nothing.

classmethod get_options()

Return the list of parameters associated with the auth plugin.

This list may be used to generate CLI or config arguments.

Returns
A list of Param objects describing available plugin parameters.

Return type
List

keystoneclient.auth.identity.**V2Password**

alias of *Password*

keystoneclient.auth.identity.**V2Token**

alias of *Token*

keystoneclient.auth.identity.**V3Password**

alias of *Password*

keystoneclient.auth.identity.**V3Token**

alias of *Token*

Submodules

keystoneclient.auth.base module

class keystoneclient.auth.base.**BaseAuthPlugin**

Bases: object

The basic structure of an authentication plugin.

get_connection_params(*session*, ***kwargs*)

Return any additional connection parameters required for the plugin.

Parameters
session (keystoneclient.session.Session) The session object that the auth_plugin belongs to.

Returns
Headers that are set to authenticate a message or None for failure. Note that when checking this value that the empty dict is a valid, non-failure response.

Return type
dict

get_endpoint(*session*, ***kwargs*)

Return an endpoint for the client.

There are no required keyword arguments to `get_endpoint` as a plugin implementation should use best effort with the information available to determine the endpoint. However there are certain standard options that will be generated by the clients and should be used by plugins:

- `service_type`: what sort of service is required.
- `service_name`: the name of the service in the catalog.
- `interface`: what visibility the endpoint should have.
- `region_name`: the region the endpoint exists in.

Parameters
session (keystoneclient.session.Session) The session object that the auth_plugin belongs to.

Returns
The base URL that will be used to talk to the required service or None if not available.

Return type
string

get_headers(*session*, ***kwargs*)

Fetch authentication headers for message.

This is a more generalized replacement of the older get_token to allow plugins to specify different or additional authentication headers to the OpenStack standard X-Auth-Token header.

How the authentication headers are obtained is up to the plugin. If the headers are still valid they may be re-used, retrieved from cache or the plugin may invoke an authentication request against a server.

The default implementation of get_headers calls the *get_token* method to enable older style plugins to continue functioning unchanged. Subclasses should feel free to completely override this function to provide the headers that they want.

There are no required kwargs. They are passed directly to the auth plugin and they are implementation specific.

Returning None will indicate that no token was able to be retrieved and that authorization was a failure. Adding no authentication data can be achieved by returning an empty dictionary.

Parameters

session (keystoneclient.session.Session) The session object that the auth_plugin belongs to.

Returns

Headers that are set to authenticate a message or None for failure. Note that when checking this value that the empty dict is a valid, non-failure response.

Return type

dict

`classmethod get_options()`

Return the list of parameters associated with the auth plugin.

This list may be used to generate CLI or config arguments.

Returns

A list of Param objects describing available plugin parameters.

Return type

List

`get_project_id(session, **kwargs)`

Return the project id that we are authenticated to.

Wherever possible the project id should be inferred from the token however there are certain URLs and other places that require access to the currently authenticated project id.

Parameters

session (keystoneclient.session.Session) A session object so the plugin can make HTTP calls.

Returns

A project identifier or None if one is not available.

Return type

str

`get_token(session, **kwargs)`

Obtain a token.

How the token is obtained is up to the plugin. If it is still valid it may be re-used, retrieved from cache or invoke an authentication request against a server.

There are no required kwargs. They are passed directly to the auth plugin and they are implementation specific.

Returning None will indicate that no token was able to be retrieved.

This function is misplaced as it should only be required for auth plugins that use the X-Auth-Token header. However due to the way plugins evolved this method is required and often called to trigger an authentication request on a new plugin.

When implementing a new plugin it is advised that you implement this method, however if you dont require the X-Auth-Token header override the *get_headers* method instead.

Parameters
session (keystoneclient.session.Session) A session object so the plugin can make HTTP calls.

Returns
A token to use.

Return type
string

`get_user_id`(*session*, ***kwargs*)

Return a unique user identifier of the plugin.

Wherever possible the user id should be inferred from the token however there are certain URLs and other places that require access to the currently authenticated user id.

Parameters
session (keystoneclient.session.Session) A session object so the plugin can make HTTP calls.

Returns
A user identifier or None if one is not available.

Return type
str

`invalidate`()

Invalidate the current authentication data.

This should result in fetching a new token on next call.

A plugin may be invalidated if an Unauthorized HTTP response is returned to indicate that the token may have been revoked or is otherwise now invalid.

Returns
True if there was something that the plugin did to invalidate. This means that it makes sense to try again. If nothing happens returns False to indicate give up.

Return type
bool

`classmethod load_from_argparse_arguments`(*namespace*, ***kwargs*)

Load a specific plugin object from an argparse result.

Convert the results of a parse into the specified plugin.

Parameters
namespace (*argparse.Namespace*) The result from CLI parsing.

Returns
An auth plugin, or None if a name is not provided.

Return type
keystoneclient.auth.BaseAuthPlugin

classmethod load_from_conf_options(*conf*, *group*, ***kwargs*)

Load the plugin from a CONF object.

Convert the options already registered into a real plugin.

Parameters

- **conf** (*oslo_config.cfg.ConfigOpts*) A config object.
- **group** (*string*) The group name that options should be read from.

Returns
An authentication Plugin.

Return type
keystoneclient.auth.BaseAuthPlugin

classmethod load_from_options(***kwargs*)

Create a plugin from the arguments retrieved from get_options.

A client can override this function to do argument validation or to handle differences between the registered options and what is required to create the plugin.

classmethod load_from_options_getter(*getter*, ***kwargs*)

Load a plugin from a getter function returning appropriate values.

To handle cases other than the provided CONF and CLI loading you can specify a custom loader function that will be queried for the option value.

The getter is a function that takes one value, an `oslo_config.cfg.Opt` and returns a value to load with.

Parameters
getter (*callable*) A function that returns a value for the given opt.

Returns
An authentication Plugin.

Return type
keystoneclient.auth.BaseAuthPlugin

classmethod register_argparse_arguments(*parser*)

Register the CLI options provided by a specific plugin.

Given a plugin class convert its options into argparse arguments and add them to a parser.

Parameters
parser (*argparse.ArgumentParser*) the parser to attach argparse options.

classmethod register_conf_options(*conf*, *group*)

Register the oslo_config options that are needed for a plugin.

Parameters

- **conf** (*oslo_config.cfg.ConfigOpts*) A config object.
- **group** (*string*) The group name that options should be read from.

keystoneclient.auth.base.**get_available_plugin_classes**()

Retrieve all the plugin classes available on the system.

Returns
A dict with plugin entrypoint name as the key and the plugin class as the value.

Return type
dict

keystoneclient.auth.base.**get_available_plugin_names**()

Get the names of all the plugins that are available on the system.

This is particularly useful for help and error text to prompt a user for example what plugins they may specify.

Returns
A list of names.

Return type
frozenset

keystoneclient.auth.base.**get_plugin_class**(*name*)

Retrieve a plugin class by its entrypoint name.

Parameters
name (*str*) The name of the object to get.

Returns
An auth plugin class.

Return type
keystoneclient.auth.BaseAuthPlugin

Raises
keystoneclient.exceptions.NoMatchingPlugin if a plugin cannot be created.

keystoneclient.auth.cli module

keystoneclient.auth.cli.**load_from_argparse_arguments**(*namespace*, ***kwargs*)

Retrieve the created plugin from the completed argparse results.

Loads and creates the auth plugin from the information parsed from the command line by argparse.

Parameters
namespace (*Namespace*) The result from CLI parsing.

Returns
An auth plugin, or None if a name is not provided.

Return type
keystoneclient.auth.BaseAuthPlugin

Raises
keystoneclient.exceptions.NoMatchingPlugin if a plugin cannot be created.

keystoneclient.auth.cli.**register_argparse_arguments**(*parser*, *argv*, *default=None*)

Register CLI options needed to create a plugin.

The function inspects the provided arguments so that it can also register the options required for that specific plugin if available.

Parameters

- **argparse.ArgumentParser** the parser to attach argparse options to.
- **argv** (*List*) the arguments provided to the application.
- **default** (*str/class*) a default plugin name or a plugin object to use if one isnt specified by the CLI. default: None.

Returns

The plugin class that will be loaded or None if not provided.

Return type

keystoneclient.auth.BaseAuthPlugin

Raises

keystoneclient.exceptions.NoMatchingPlugin if a plugin cannot be created.

keystoneclient.auth.conf module

keystoneclient.auth.conf.**get_common_conf_options**()

Get the oslo_config options common for all auth plugins.

These may be useful without being registered for config file generation or to manipulate the options before registering them yourself.

The options that are set are:

auth_plugin

The name of the plugin to load.

auth_section

The config file section to load options from.

Returns

A list of oslo_config options.

keystoneclient.auth.conf.**get_plugin_options**(*name*)

Get the oslo_config options for a specific plugin.

This will be the list of config options that is registered and loaded by the specified plugin.

Returns

A list of oslo_config options.

keystoneclient.auth.conf.**load_from_conf_options**(*conf*, *group*, ***kwargs*)

Load a plugin from an oslo_config CONF object.

Each plugin will register their own required options and so there is no standard list and the plugin should be consulted.

The base options should have been registered with register_conf_options before this function is called.

Parameters

- **conf** (*oslo_config.cfg.ConfigOpts*) A conf object.
- **group** (*string*) The group name that options should be read from.

Returns
An authentication Plugin or None if a name is not provided

Return type
keystoneclient.auth.BaseAuthPlugin

Raises
keystoneclient.exceptions.NoMatchingPlugin if a plugin cannot be created.

keystoneclient.auth.conf.**register_conf_options**(*conf*, *group*)

Register the oslo_config options that are needed for a plugin.

This only registers the basic options shared by all plugins. Options that are specific to a plugin are loaded just before they are read.

The defined options are:

- **auth_plugin: the name of the auth plugin that will be used for** authentication.
- **auth_section: the group from which further auth plugin options should be** taken. If section is not provided then the auth plugin options will be taken from the same group as provided in the parameters.

Parameters

- **conf** (*oslo_config.cfg.ConfigOpts*) config object to register with.
- **group** (*string*) The ini group to register options in.

keystoneclient.auth.token_endpoint module

class keystoneclient.auth.token_endpoint.**Token**(*endpoint*, *token*)

Bases: *BaseAuthPlugin*

A provider that will always use the given token and endpoint.

This is really only useful for testing and in certain CLI cases where you have a known endpoint and admin token that you want to use.

get_endpoint(*session*, ***kwargs*)

Return the supplied endpoint.

Using this plugin the same endpoint is returned regardless of the parameters passed to the plugin.

classmethod get_options()

Return the list of parameters associated with the auth plugin.

This list may be used to generate CLI or config arguments.

Returns
A list of Param objects describing available plugin parameters.

Return type
List

get_token(*session*)

Obtain a token.

How the token is obtained is up to the plugin. If it is still valid it may be re-used, retrieved from cache or invoke an authentication request against a server.

There are no required kwargs. They are passed directly to the auth plugin and they are implementation specific.

Returning None will indicate that no token was able to be retrieved.

This function is misplaced as it should only be required for auth plugins that use the X-Auth-Token header. However due to the way plugins evolved this method is required and often called to trigger an authentication request on a new plugin.

When implementing a new plugin it is advised that you implement this method, however if you dont require the X-Auth-Token header override the *get_headers* method instead.

Parameters
session (keystoneclient.session.Session) A session object so the plugin can make HTTP calls.

Returns
A token to use.

Return type
string

Module contents

class keystoneclient.auth.**BaseAuthPlugin**

Bases: object

The basic structure of an authentication plugin.

get_connection_params(*session*, ***kwargs*)

Return any additional connection parameters required for the plugin.

Parameters
session (keystoneclient.session.Session) The session object that the auth_plugin belongs to.

Returns
Headers that are set to authenticate a message or None for failure. Note that when checking this value that the empty dict is a valid, non-failure response.

Return type
dict

`get_endpoint`(*session*, ***kwargs*)

Return an endpoint for the client.

There are no required keyword arguments to `get_endpoint` as a plugin implementation should use best effort with the information available to determine the endpoint. However there are certain standard options that will be generated by the clients and should be used by plugins:

- `service_type`: what sort of service is required.
- `service_name`: the name of the service in the catalog.
- `interface`: what visibility the endpoint should have.
- `region_name`: the region the endpoint exists in.

Parameters
session (keystoneclient.session.Session) The session object that the auth_plugin belongs to.

Returns
The base URL that will be used to talk to the required service or None if not available.

Return type
string

`get_headers`(*session*, ***kwargs*)

Fetch authentication headers for message.

This is a more generalized replacement of the older get_token to allow plugins to specify different or additional authentication headers to the OpenStack standard X-Auth-Token header.

How the authentication headers are obtained is up to the plugin. If the headers are still valid they may be re-used, retrieved from cache or the plugin may invoke an authentication request against a server.

The default implementation of get_headers calls the *get_token* method to enable older style plugins to continue functioning unchanged. Subclasses should feel free to completely override this function to provide the headers that they want.

There are no required kwargs. They are passed directly to the auth plugin and they are implementation specific.

Returning None will indicate that no token was able to be retrieved and that authorization was a failure. Adding no authentication data can be achieved by returning an empty dictionary.

Parameters
session (keystoneclient.session.Session) The session object that the auth_plugin belongs to.

Returns
Headers that are set to authenticate a message or None for failure. Note that when checking this value that the empty dict is a valid, non-failure response.

Return type
dict

`classmethod get_options()`

Return the list of parameters associated with the auth plugin.

This list may be used to generate CLI or config arguments.

Returns
A list of Param objects describing available plugin parameters.

Return type
List

`get_project_id(session, **kwargs)`

Return the project id that we are authenticated to.

Wherever possible the project id should be inferred from the token however there are certain URLs and other places that require access to the currently authenticated project id.

Parameters
session (keystoneclient.session.Session) A session object so the plugin can make HTTP calls.

Returns
A project identifier or None if one is not available.

Return type
str

`get_token(session, **kwargs)`

Obtain a token.

How the token is obtained is up to the plugin. If it is still valid it may be re-used, retrieved from cache or invoke an authentication request against a server.

There are no required kwargs. They are passed directly to the auth plugin and they are implementation specific.

Returning None will indicate that no token was able to be retrieved.

This function is misplaced as it should only be required for auth plugins that use the X-Auth-Token header. However due to the way plugins evolved this method is required and often called to trigger an authentication request on a new plugin.

When implementing a new plugin it is advised that you implement this method, however if you dont require the X-Auth-Token header override the *get_headers* method instead.

Parameters
session (keystoneclient.session.Session) A session object so the plugin can make HTTP calls.

Returns
A token to use.

Return type
string

get_user_id(*session*, ***kwargs*)

Return a unique user identifier of the plugin.

Wherever possible the user id should be inferred from the token however there are certain URLs and other places that require access to the currently authenticated user id.

Parameters

session (keystoneclient.session.Session) A session object so the plugin can make HTTP calls.

Returns

A user identifier or None if one is not available.

Return type

str

invalidate()

Invalidate the current authentication data.

This should result in fetching a new token on next call.

A plugin may be invalidated if an Unauthorized HTTP response is returned to indicate that the token may have been revoked or is otherwise now invalid.

Returns

True if there was something that the plugin did to invalidate. This means that it makes sense to try again. If nothing happens returns False to indicate give up.

Return type

bool

classmethod load_from_argparse_arguments(*namespace*, ***kwargs*)

Load a specific plugin object from an argparse result.

Convert the results of a parse into the specified plugin.

Parameters

namespace (*argparse.Namespace*) The result from CLI parsing.

Returns

An auth plugin, or None if a name is not provided.

Return type

keystoneclient.auth.BaseAuthPlugin

classmethod load_from_conf_options(*conf*, *group*, ***kwargs*)

Load the plugin from a CONF object.

Convert the options already registered into a real plugin.

Parameters

- **conf** (*oslo_config.cfg.ConfigOpts*) A config object.
- **group** (*string*) The group name that options should be read from.

Returns

An authentication Plugin.

Return type

keystoneclient.auth.BaseAuthPlugin

classmethod load_from_options(***kwargs*)

Create a plugin from the arguments retrieved from get_options.

A client can override this function to do argument validation or to handle differences between the registered options and what is required to create the plugin.

classmethod load_from_options_getter(*getter*, ***kwargs*)

Load a plugin from a getter function returning appropriate values.

To handle cases other than the provided CONF and CLI loading you can specify a custom loader function that will be queried for the option value.

The getter is a function that takes one value, an `oslo_config.cfg.Opt` and returns a value to load with.

Parameters
getter (*callable*) A function that returns a value for the given opt.

Returns
An authentication Plugin.

Return type
keystoneclient.auth.BaseAuthPlugin

classmethod register_argparse_arguments(*parser*)

Register the CLI options provided by a specific plugin.

Given a plugin class convert its options into argparse arguments and add them to a parser.

Parameters
parser (*argparse.ArgumentParser*) the parser to attach argparse options.

classmethod register_conf_options(*conf*, *group*)

Register the oslo_config options that are needed for a plugin.

Parameters

- **conf** (*oslo_config.cfg.ConfigOpts*) A config object.
- **group** (*string*) The group name that options should be read from.

keystoneclient.auth.**get_available_plugin_classes**()

Retrieve all the plugin classes available on the system.

Returns
A dict with plugin entrypoint name as the key and the plugin class as the value.

Return type
dict

keystoneclient.auth.**get_available_plugin_names**()

Get the names of all the plugins that are available on the system.

This is particularly useful for help and error text to prompt a user for example what plugins they may specify.

Returns
A list of names.

Return type
frozenset

keystoneclient.auth.**get_common_conf_options**()

Get the oslo_config options common for all auth plugins.

These may be useful without being registered for config file generation or to manipulate the options before registering them yourself.

The options that are set are:

auth_plugin
The name of the plugin to load.

auth_section
The config file section to load options from.

Returns
A list of oslo_config options.

keystoneclient.auth.**get_plugin_class**(*name*)

Retrieve a plugin class by its entrypoint name.

Parameters
name (*str*) The name of the object to get.

Returns
An auth plugin class.

Return type
keystoneclient.auth.BaseAuthPlugin

Raises
keystoneclient.exceptions.NoMatchingPlugin if a plugin cannot be created.

keystoneclient.auth.**get_plugin_options**(*name*)

Get the oslo_config options for a specific plugin.

This will be the list of config options that is registered and loaded by the specified plugin.

Returns
A list of oslo_config options.

keystoneclient.auth.**load_from_argparse_arguments**(*namespace*, ***kwargs*)

Retrieve the created plugin from the completed argparse results.

Loads and creates the auth plugin from the information parsed from the command line by argparse.

Parameters
namespace (*Namespace*) The result from CLI parsing.

Returns
An auth plugin, or None if a name is not provided.

Return type
keystoneclient.auth.BaseAuthPlugin

Raises
keystoneclient.exceptions.NoMatchingPlugin if a plugin cannot be created.

keystoneclient.auth.**load_from_conf_options**(*conf*, *group*, ***kwargs*)

Load a plugin from an oslo_config CONF object.

Each plugin will register their own required options and so there is no standard list and the plugin should be consulted.

The base options should have been registered with register_conf_options before this function is called.

Parameters

- **conf** (*oslo_config.cfg.ConfigOpts*) A conf object.
- **group** (*string*) The group name that options should be read from.

Returns
An authentication Plugin or None if a name is not provided

Return type
keystoneclient.auth.BaseAuthPlugin

Raises
keystoneclient.exceptions.NoMatchingPlugin if a plugin cannot be created.

keystoneclient.auth.**register_argparse_arguments**(*parser*, *argv*, *default=None*)

Register CLI options needed to create a plugin.

The function inspects the provided arguments so that it can also register the options required for that specific plugin if available.

Parameters

- **argparse.ArgumentParser** the parser to attach argparse options to.
- **argv** (*List*) the arguments provided to the application.
- **default** (*str/class*) a default plugin name or a plugin object to use if one isnt specified by the CLI. default: None.

Returns
The plugin class that will be loaded or None if not provided.

Return type
keystoneclient.auth.BaseAuthPlugin

Raises
keystoneclient.exceptions.NoMatchingPlugin if a plugin cannot be created.

keystoneclient.auth.**register_conf_options**(*conf*, *group*)

Register the oslo_config options that are needed for a plugin.

This only registers the basic options shared by all plugins. Options that are specific to a plugin are loaded just before they are read.

The defined options are:

- **auth_plugin: the name of the auth plugin that will be used for** authentication.

- **auth_section: the group from which further auth plugin options should be**
  taken. If section is not provided then the auth plugin options will be taken from the same group as provided in the parameters.

Parameters

- **conf** (*oslo_config.cfg.ConfigOpts*) config object to register with.
- **group** (*string*) The ini group to register options in.

keystoneclient.common package

Submodules

keystoneclient.common.cms module

Certificate signing functions.

Call set_subprocess() with the subprocess module. Either Pythons subprocess or eventlet.green.subprocess can be used.

If set_subprocess() is not called, this module will pick Pythons subprocess or eventlet.green.subprocess based on if os module is patched by eventlet.

class keystoneclient.common.cms.**OpensslCmsExitStatus**

Bases: object

COMMAND_OPTIONS_PARSING_ERROR = 1

CREATE_CMS_READ_MIME_ERROR = 3

INPUT_FILE_READ_ERROR = 2

SUCCESS = 0

keystoneclient.common.cms.**cms_hash_token**(*token_id*, *mode='md5'*)

Hash PKI tokens.

return: for asn1 or pkiz tokens, returns the hash of the passed in token
otherwise, returns what it was passed in.

keystoneclient.common.cms.**cms_sign_data**(*data_to_sign*, *signing_cert_file_name*, *signing_key_file_name*, *outform='PEM'*, *message_digest='sha256'*)

Use OpenSSL to sign a document.

Produces a Base64 encoding of a DER formatted CMS Document http://en.wikipedia.org/wiki/Cryptographic_Message_Syntax

Parameters

- **data_to_sign** data to sign
- **signing_cert_file_name** path to the X509 certificate containing the public key associated with the private key used to sign the data
- **signing_key_file_name** path to the private key used to sign the data

- **outform** Format for the signed document PKIZ_CMS_FORM or PKI_ASN1_FORM
- **message_digest** Digest algorithm to use when signing or resigning

keystoneclient.common.cms.**cms_sign_text**(*data_to_sign*, *signing_cert_file_name*, *signing_key_file_name*, *message_digest='sha256'*)

keystoneclient.common.cms.**cms_sign_token**(*text*, *signing_cert_file_name*, *signing_key_file_name*, *message_digest='sha256'*)

keystoneclient.common.cms.**cms_to_token**(*cms_text*)

Convert a CMS-signed token in PEM format to a custom URL-safe format.

The conversion consists of replacing / char in the PEM-formatted token with the - char and doing other such textual replacements to make the result marshallable via HTTP. The return value can thus be used as the value of a HTTP header such as X-Auth-Token.

This ad-hoc conversion is an unfortunate oversight since the returned value now does not conform to any of the standard variants of base64 encoding. It would have been better to use base64url encoding (either on the PEM formatted text or, perhaps even better, on the inner CMS-signed binary value without any PEM formatting). In any case, the same conversion is done in reverse in the other direction (for token verification), so there are no correctness issues here. Note that the non-standard encoding of the token will be preserved so as to not break backward compatibility.

The conversion issue is detailed by the code author in a blog post at http://adam.younglogic.com/2014/02/compressed-tokens/.

keystoneclient.common.cms.**cms_verify**(*formatted*, *signing_cert_file_name*, *ca_file_name*, *inform='PEM'*)

Verify the signature of the contents IAW CMS syntax.

Raises

- subprocess.CalledProcessError
- *keystoneclient.exceptions.CertificateConfigError* if certificate is not configured properly.

keystoneclient.common.cms.**is_ans1_token**(*token*)

Deprecated.

This function is deprecated as of the 1.7.0 release in favor of *is_asn1_token()* and may be removed in the 2.0.0 release.

keystoneclient.common.cms.**is_asn1_token**(*token*)

Determine if a token appears to be PKI-based.

thx to ayoung for sorting this out.

base64 decoded hex representation of MII is 3082:

```
In [3]: binascii.hexlify(base64.b64decode('MII='))
Out[3]: '3082'
```

re: http://www.itu.int/ITU-T/studygroups/com17/languages/X.690-0207.pdf

```
pg4:  For tags from 0 to 30 the first octet is the identfier
pg10: Hex 30 means sequence, followed by the length of that sequence.
pg5:  Second octet is the length octet
      first bit indicates short or long form, next 7 bits encode the
      number of subsequent octets that make up the content length octets
      as an unsigned binary int

      82 = 10000010 (first bit indicates long form)
      0000010 = 2 octets of content length
      so read the next 2 octets to get the length of the content.
```

In the case of a very large content length there could be a requirement to have more than 2 octets to designate the content length, therefore requiring us to check for MIM, MIQ, etc.

```
In [4]: base64.b64encode(binascii.a2b_hex('3083'))
Out[4]: 'MIM='
In [5]: base64.b64encode(binascii.a2b_hex('3084'))
Out[5]: 'MIQ='
Checking for MI would become invalid at 16 octets of content length
10010000 = 90
In [6]: base64.b64encode(binascii.a2b_hex('3090'))
Out[6]: 'MJA='
Checking for just M is insufficient
```

But we will only check for MII: Max length of the content using 2 octets is 3FFF or 16383.

Its not practical to support a token of this length or greater in http therefore, we will check for MII only and ignore the case of larger tokens

keystoneclient.common.cms.**is_pkiz**(*token_text*)

Determine if a token is PKIZ.

Checks if the string has the prefix that indicates it is a Crypto Message Syntax, Z compressed token.

keystoneclient.common.cms.**pkiz_sign**(*text*, *signing_cert_file_name*, *signing_key_file_name*, *compression_level=6*, *message_digest='sha256'*)

keystoneclient.common.cms.**pkiz_uncompress**(*signed_text*)

keystoneclient.common.cms.**pkiz_verify**(*signed_text*, *signing_cert_file_name*, *ca_file_name*)

keystoneclient.common.cms.**set_subprocess**(*_subprocess=None*)

Set subprocess module to use.

The subprocess could be eventlet.green.subprocess if using eventlet, or Pythons subprocess otherwise.

keystoneclient.common.cms.**token_to_cms**(*signed_text*)

Convert a custom formatted token to a PEM-formatted token.

See documentation for cms_to_token() for details on the custom formatting.

keystoneclient.common.cms.**verify_token**(*token*, *signing_cert_file_name*, *ca_file_name*)

Module contents

keystoneclient.contrib package

Subpackages

keystoneclient.contrib.auth package

Subpackages

keystoneclient.contrib.auth.v3 package

Submodules

keystoneclient.contrib.auth.v3.oidc module

class keystoneclient.contrib.auth.v3.oidc.**OidcPassword**(*auth_url*, *identity_provider*, *protocol*, *username*, *password*, *client_id*, *client_secret*, *access_token_endpoint*, *scope='profile'*, *grant_type='password'*)

Bases: *FederatedBaseAuth*

Implement authentication plugin for OpenID Connect protocol.

OIDC or OpenID Connect is a protocol for federated authentication.

The OpenID Connect specification can be found at:: `http://openid.net/specs/openid-connect-core-1_0.html`

classmethod get_options()

Return the list of parameters associated with the auth plugin.

This list may be used to generate CLI or config arguments.

Returns
A list of Param objects describing available plugin parameters.

Return type
List

get_unscoped_auth_ref(*session*)

Authenticate with OpenID Connect and get back claims.

This is a multi-step process. First an access token must be retrieved, to do this, the username and password, the OpenID Connect client ID and secret, and the access token endpoint must be known.

Secondly, we then exchange the access token upon accessing the protected Keystone endpoint (federated auth URL). This will trigger the OpenID Connect Provider to perform a user introspection and retrieve information (specified in the scope) about the user in the form of an OpenID Connect Claim. These claims will be sent to Keystone in the form of environment variables.

Parameters
session (keystoneclient.session.Session) a session object to send out HTTP requests.

Returns
a token data representation

Return type
keystoneclient.access.AccessInfo

property password

Deprecated as of the 1.7.0 release.

It may be removed in the 2.0.0 release.

property username

Deprecated as of the 1.7.0 release.

It may be removed in the 2.0.0 release.

keystoneclient.contrib.auth.v3.saml2 module

class keystoneclient.contrib.auth.v3.saml2.**ADFSUnscopedToken**(*auth_url*, *identity_provider*, *identity_provider_url*, *service_provider_endpoint*, *username*, *password*, ***kwargs*)

Bases: _BaseSAMLPlugin

Authentication plugin for Microsoft ADFS2.0 IdPs.

Parameters

- **auth_url** (*string*) URL of the Identity Service
- **identity_provider** (*string*) name of the Identity Provider the client will authenticate against. This parameter will be used to build a dynamic URL used to obtain unscoped OpenStack token.
- **identity_provider_url** (*string*) An Identity Provider URL, where the SAML2 authentication request will be sent.
- **service_provider_endpoint** (*string*) Endpoint where an assertion is being sent, for instance: https://host.domain/Shibboleth.sso/ADFS
- **username** (*string*) Users login
- **password** (*string*) Users password

ADFS_ASSERTION_XPATH = '/s:Envelope/s:Body/ t:RequestSecurityTokenResponseCollection/t:RequestSecurityTokenResponse'

ADFS_TOKEN_NAMESPACES = {'s': 'http://www.w3.org/2003/05/soap-envelope', 't': 'http://docs.oasis-open.org/ws-sx/ws-trust/200512'}

```
DEFAULT_ADFS_TOKEN_EXPIRATION = 120

HEADER_SOAP = {'Content-Type':  'application/soap+xml; charset=utf-8'}

HEADER_X_FORM = {'Content-Type':  'application/x-www-form-urlencoded'}

NAMESPACES = {'a':  'http://www.w3.org/2005/08/addressing', 's':
'http://www.w3.org/2003/05/soap-envelope', 'u':  'http://docs.oasis-open.
org/wss/2004/01/oasis-200401-wss-wssecurity-utility-1.0.xsd'}
```

get_auth_ref(*session*, ***kwargs*)

Obtain a token from an OpenStack Identity Service.

This method is overridden by the various token version plugins.

This method should not be called independently and is expected to be invoked via the do_authenticate() method.

This method will be invoked if the AccessInfo object cached by the plugin is not valid. Thus plugins should always fetch a new AccessInfo when invoked. If you are looking to just retrieve the current auth data then you should use get_access().

Parameters

session (keystoneclient.session.Session) A session object that can be used for communication.

Raises

- *keystoneclient.exceptions.InvalidResponse* The response returned wasnt appropriate.
- *keystoneclient.exceptions.HttpError* An error from an invalid HTTP response.

Returns

Token access information.

Return type

keystoneclient.access.AccessInfo

classmethod get_options()

Return the list of parameters associated with the auth plugin.

This list may be used to generate CLI or config arguments.

Returns

A list of Param objects describing available plugin parameters.

Return type

List

property password

Deprecated as of the 1.7.0 release.

It may be removed in the 2.0.0 release.

property username

Deprecated as of the 1.7.0 release.

It may be removed in the 2.0.0 release.

class keystoneclient.contrib.auth.v3.saml2.**Saml2ScopedToken**(*auth_url*, *token*, ***kwargs*)

Bases: *Token*

Class for scoping unscoped saml2 token.

class keystoneclient.contrib.auth.v3.saml2.**Saml2ScopedTokenMethod**(***kwargs*)

Bases: *TokenMethod*

get_auth_data(*session*, *auth*, *headers*, ***kwargs*)

Build and return request body for token scoping step.

class keystoneclient.contrib.auth.v3.saml2.**Saml2UnscopedToken**(*auth_url*, *identity_provider*, *identity_provider_url*, *username*, *password*, ***kwargs*)

Bases: _BaseSAMLPlugin

Implement authentication plugin for SAML2 protocol.

ECP stands for *Enhanced Client or Proxy* and is a SAML2 extension for federated authentication where a transportation layer consists of HTTP protocol and XML SOAP messages.

Read for more information on ECP.

Reference the SAML2 ECP specification.

Currently only HTTPBasicAuth mechanism is available for the IdP authenication.

Parameters

- **auth_url** (*string*) URL of the Identity Service
- **identity_provider** (*string*) name of the Identity Provider the client will authenticate against. This parameter will be used to build a dynamic URL used to obtain unscoped OpenStack token.
- **identity_provider_url** (*string*) An Identity Provider URL, where the SAML2 authn request will be sent.
- **username** (*string*) Users login
- **password** (*string*) Users password

ECP_IDP_CONSUMER_URL = '/S:Envelope/S:Header/ecp:Response/@AssertionConsumerServiceURL'

ECP_RELAY_STATE = '//ecp:RelayState'

ECP_SAML2_NAMESPACES = {'S': 'http://schemas.xmlsoap.org/soap/envelope/', 'ecp': 'urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp', 'paos': 'urn:liberty:paos:2003-08'}

ECP_SERVICE_PROVIDER_CONSUMER_URL = '/S:Envelope/S:Header/paos:Request/@responseConsumerURL'

```
ECP_SP_EMPTY_REQUEST_HEADERS = {'Accept':  'text/html,
application/vnd.paos+xml', 'PAOS': 'ver="urn:liberty:paos:2003-08";
"urn:oasis:names:tc:SAML:2.0:profiles:SSO:ecp"'}

ECP_SP_SAML2_REQUEST_HEADERS = {'Content-Type':
'application/vnd.paos+xml'}

SAML2_HEADER_INDEX = 0

SOAP_FAULT = '\n <S:Envelope
xmlns:S="http://schemas.xmlsoap.org/soap/envelope/">\n <S:Body>\n
<S:Fault>\n <faultcode>S:Server</faultcode>\n
<faultstring>responseConsumerURL from SP and\n assertionConsumerServiceURL
from IdP do not match\n </faultstring>\n </S:Fault>\n </S:Body>\n
</S:Envelope>\n '
```

get_auth_ref(*session*, ***kwargs*)

Authenticate via SAML2 protocol and retrieve unscoped token.

This is a multi-step process where a client does federated authn receives an unscoped token.

Federated authentication utilizing SAML2 Enhanced Client or Proxy extension. See `Saml2UnscopedToken_get_unscoped_token()` for more information on that step. Upon successful authentication and assertion mapping an unscoped token is returned and stored within the plugin object for further use.

:param session : a session object to send out HTTP requests. :type session: keystoneclient.session.Session

Returns
an object with scoped tokens id and unscoped token json included.

Return type
keystoneclient.access.AccessInfoV3

property password

Deprecated as of the 1.7.0 release.

It may be removed in the 2.0.0 release.

property username

Deprecated as of the 1.7.0 release.

It may be removed in the 2.0.0 release.

class keystoneclient.contrib.auth.v3.saml2.**Saml2UnscopedTokenAuthMethod**(***kwargs*)

Bases: *AuthMethod*

get_auth_data(*session*, *auth*, *headers*, ***kwargs*)

Return the authentication section of an auth plugin.

Parameters

- **session** (keystoneclient.session.Session) The communication session.
- **auth** (base.Auth) The auth plugin calling the method.

- **headers** (*dict*) The headers that will be sent with the auth request if a plugin needs to add to them.

Returns
The identifier of this plugin and a dict of authentication data for the auth type.

Return type
tuple(string, dict)

Module contents

Module contents

keystoneclient.contrib.ec2 package

Submodules

keystoneclient.contrib.ec2.utils module

class keystoneclient.contrib.ec2.utils.**Ec2Signer**(*secret_key*)

Bases: object

Utility class for EC2 signing of request.

This allows a request to be signed with an AWS style signature, which can then be used for authentication via the keystone ec2 authentication extension.

generate(*credentials*)

Generate auth string according to what SignatureVersion is given.

Module contents

Module contents

keystoneclient.generic package

Submodules

keystoneclient.generic.client module

class keystoneclient.generic.client.**Client**(*endpoint=None*, ***kwargs*)

Bases: *HTTPClient*

Client for the OpenStack Keystone pre-version calls API.

Parameters

- **endpoint** (*string*) A user-supplied endpoint URL for the keystone service.
- **timeout** (*integer*) Allows customization of the timeout for client http requests. (optional)

Example:

```
>>> from keystoneclient.generic import client
>>> root = client.Client(auth_url=KEYSTONE_URL)
>>> versions = root.discover()
...
>>> from keystoneclient.v2_0 import client as v2client
>>> keystone = v2client.Client(auth_url=versions['v2.0']['url'])
...
>>> user = keystone.users.get(USER_ID)
>>> user.delete()
```

discover(*url=None*)

Discover Keystone servers and return API versions supported.

Parameters

url optional url to test (without version)

Returns:

```
{
    'message': 'Keystone found at http://127.0.0.1:5000/',
    'v2.0': {
        'status': 'beta',
        'url': 'http://127.0.0.1:5000/v2.0/',
        'id': 'v2.0'
    },
}
```

discover_extensions(*url=None*)

Discover Keystone extensions supported.

Parameters

url optional url to test (should have a version in it)

Returns:

```
{
    'message': 'Keystone extensions at http://127.0.0.1:35357/v2',
    'OS-KSEC2': 'OpenStack EC2 Credentials Extension',
}
```

Module contents

keystoneclient.v2_0 package

Submodules

keystoneclient.v2_0.certificates module

class keystoneclient.v2_0.certificates.**CertificatesManager**(*client*)

Bases: object

Manager for certificates.

get_ca_certificate()

Get CA certificate.

Returns
PEM-formatted string.

Return type
str

get_signing_certificate()

Get signing certificate.

Returns
PEM-formatted string.

Return type
str

keystoneclient.v2_0.client module

class keystoneclient.v2_0.client.**Client**(**kwargs*)

Bases: *HTTPClient*

Client for the OpenStack Keystone v2.0 API.

Parameters

- **username** (*string*) Username for authentication. (optional)
- **password** (*string*) Password for authentication. (optional)
- **token** (*string*) Token for authentication. (optional)
- **tenant_id** (*string*) Tenant id. (optional)
- **tenant_name** (*string*) Tenant name. (optional)
- **auth_url** (*string*) Keystone service endpoint for authorization.
- **region_name** (*string*) Name of a region to select when choosing an endpoint from the service catalog.
- **endpoint** (*string*) A user-supplied endpoint URL for the keystone service. Lazy-authentication is possible for API service calls if endpoint is set at instantiation.(optional)
- **timeout** (*integer*) Allows customization of the timeout for client http requests. (optional)
- **original_ip** (*string*) The original IP of the requesting user which will be sent to Keystone in a Forwarded header. (optional)
- **cert** (*string*) Path to the Privacy Enhanced Mail (PEM) file which contains the corresponding X.509 client certificate needed to established two-way SSL connection with the identity service. (optional)

- **key** (*string*) Path to the Privacy Enhanced Mail (PEM) file which contains the unencrypted client private key needed to established two-way SSL connection with the identity service. (optional)
- **cacert** (*string*) Path to the Privacy Enhanced Mail (PEM) file which contains the trusted authority X.509 certificates needed to established SSL connection with the identity service. (optional)
- **insecure** (*boolean*) Does not perform X.509 certificate validation when establishing SSL connection with identity service. default: False (optional)
- **auth_ref** (*dict*) To allow for consumers of the client to manage their own caching strategy, you may initialize a client with a previously captured auth_reference (token)
- **debug** (*boolean*) Enables debug logging of all request and responses to keystone. default False (option)

Warning: If debug is enabled, it may show passwords in plain text as a part of its output.

Warning: Constructing an instance of this class without a session is deprecated as of the 1.7.0 release and will be removed in the 2.0.0 release.

The client can be created and used like a user or in a strictly bootstrap mode. Normal operation expects a username, password, auth_url, and tenant_name or id to be provided. Other values will be lazily loaded as needed from the service catalog.

Example:

```
>>> from keystoneauth1.identity import v2
>>> from keystoneauth1 import session
>>> from keystoneclient.v2_0 import client
>>> auth = v2.Password(auth_url=KEYSTONE_URL,
...                    username=USER,
...                    password=PASS,
...                    tenant_name=TENANT_NAME)
>>> sess = session.Session(auth=auth)
>>> keystone = client.Client(session=sess)
>>> keystone.tenants.list()
...
>>> user = keystone.users.get(USER_ID)
>>> user.delete()
```

Once authenticated, you can store and attempt to re-use the authenticated token. the auth_ref property on the client returns as a dictionary-like-object so that you can export and cache it, reusing it when initiating another client:

```
>>> from keystoneauth1.identity import v2
>>> from keystoneauth1 import session
>>> from keystoneclient.v2_0 import client
```

(continues on next page)

(continued from previous page)

```
>>> auth = v2.Password(auth_url=KEYSTONE_URL,
...                    username=USER,
...                    password=PASS,
...                    tenant_name=TENANT_NAME)
>>> sess = session.Session(auth=auth)
>>> keystone = client.Client(session=sess)
>>> auth_ref = keystone.auth_ref
>>> # pickle or whatever you like here
>>> new_client = client.Client(auth_ref=auth_ref)
```

Alternatively, you can provide the administrative token configured in keystone and an endpoint to communicate with directly. See (`admin_token` in `keystone.conf`) In this case, authenticate() is not needed, and no service catalog will be loaded.

Example:

```
>>> from keystoneauth1.identity import v2
>>> from keystoneauth1 import session
>>> from keystoneclient.v2_0 import client
>>> auth = v2.Token(auth_url='http://localhost:35357/v2.0',
...                 token='12345secret7890')
>>> sess = session.Session(auth=auth)
>>> keystone = client.Client(session=sess)
>>> keystone.tenants.list()
```

get_raw_token_from_identity_service(*auth_url*, *username=None*, *password=None*, *tenant_name=None*, *tenant_id=None*, *token=None*, *project_name=None*, *project_id=None*, *trust_id=None*, ***kwargs*)

Authenticate against the v2 Identity API.

If a token is provided it will be used in preference over username and password.

Returns
access.AccessInfo if authentication was successful.

Raises
keystoneclient.exceptions.AuthorizationFailure if unable to authenticate or validate the existing authorization token

version = 'v2.0'

keystoneclient.v2_0.ec2 module

class keystoneclient.v2_0.ec2.**CredentialsManager**(*client*)

Bases: *ManagerWithFind*

create(*user_id*, *tenant_id*)

Create a new access/secret pair for the user/tenant pair.

Return type
object of type *EC2*

delete(*user_id*, *access*)

Delete an access/secret pair for a user.

get(*user_id*, *access*)

Get the access/secret pair for a given access key.

Return type
object of type *EC2*

list(*user_id*)

Get a list of access/secret pairs for a user_id.

Return type
list of *EC2*

resource_class

alias of *EC2*

class keystoneclient.v2_0.ec2.**EC2**(*manager*, *info*, *loaded=False*)

Bases: *Resource*

delete()

keystoneclient.v2_0.endpoints module

class keystoneclient.v2_0.endpoints.**Endpoint**(*manager*, *info*, *loaded=False*)

Bases: *Resource*

Represents a Keystone endpoint.

class keystoneclient.v2_0.endpoints.**EndpointManager**(*client*)

Bases: *ManagerWithFind*

Manager class for manipulating Keystone endpoints.

create(*region*, *service_id*, *publicurl*, *adminurl=None*, *internalurl=None*)

Create a new endpoint.

delete(*id*)

Delete an endpoint.

list()

List all available endpoints.

resource_class

alias of *Endpoint*

keystoneclient.v2_0.extensions module

class keystoneclient.v2_0.extensions.**Extension**(*manager*, *info*, *loaded=False*)

Bases: *Resource*

Represents an Identity API extension.

class keystoneclient.v2_0.extensions.**ExtensionManager**(*client*)

Bases: *ManagerWithFind*

Manager class for listing Identity API extensions.

list()

List all available extensions.

resource_class

alias of *Extension*

keystoneclient.v2_0.roles module

class keystoneclient.v2_0.roles.**Role**(*manager*, *info*, *loaded=False*)

Bases: *Resource*

Represents a Keystone role.

delete()

class keystoneclient.v2_0.roles.**RoleManager**(*client*)

Bases: *ManagerWithFind*

Manager class for manipulating Keystone roles.

add_user_role(*user*, *role*, *tenant=None*)

Add a role to a user.

If tenant is specified, the role is added just for that tenant, otherwise the role is added globally.

create(*name*)

Create a role.

delete(*role*)

Delete a role.

get(*role*)

list()

List all available roles.

remove_user_role(*user*, *role*, *tenant=None*)

Remove a role from a user.

If tenant is specified, the role is removed just for that tenant, otherwise the role is removed from the users global roles.

resource_class

alias of *Role*

roles_for_user(*user*, *tenant=None*)

keystoneclient.v2_0.services module

class keystoneclient.v2_0.services.**Service**(*manager*, *info*, *loaded=False*)

Bases: *Resource*

Represents a Keystone service.

class keystoneclient.v2_0.services.**ServiceManager**(*client*)

Bases: *ManagerWithFind*

Manager class for manipulating Keystone services.

create(*name*, *service_type*, *description=None*)

Create a new service.

delete(*id*)

Delete a service.

get(*id*)

Retrieve a service by id.

list()

List available services.

resource_class

alias of *Service*

keystoneclient.v2_0.tenants module

class keystoneclient.v2_0.tenants.**Tenant**(*manager*, *info*, *loaded=False*)

Bases: *Resource*

Represents a Keystone tenant.

Attributes:

- id: a uuid that identifies the tenant
- name: tenant name
- description: tenant description
- enabled: boolean to indicate if tenant is enabled

add_user(*user*, *role*)

delete()

list_users()

remove_user(*user*, *role*)

update(*name=None*, *description=None*, *enabled=None*)

class keystoneclient.v2_0.tenants.**TenantManager**(*client*, *role_manager*, *user_manager*)

Bases: *ManagerWithFind*

Manager class for manipulating Keystone tenants.

add_user(*tenant*, *user*, *role*)

Add a user to a tenant with the given role.

create(*tenant_name*, *description=None*, *enabled=True*, ***kwargs*)

Create a new tenant.

delete(*tenant*)

Delete a tenant.

get(*tenant_id*)

list(*limit=None*, *marker=None*)

Get a list of tenants.

Parameters

- **limit** (*integer*) maximum number to return. (optional)
- **marker** (*string*) use when specifying a limit and making multiple calls for querying. (optional)

Return type

list of *Tenant*

list_users(*tenant*)

List users for a tenant.

remove_user(*tenant*, *user*, *role*)

Remove the specified role from the user on the tenant.

resource_class

alias of *Tenant*

update(*tenant_id*, *tenant_name=None*, *description=None*, *enabled=None*, ***kwargs*)

Update a tenant with a new name and description.

keystoneclient.v2_0.tokens module

class keystoneclient.v2_0.tokens.**Token**(*manager*, *info*, *loaded=False*)

Bases: *Resource*

property expires

property id

property tenant

class keystoneclient.v2_0.tokens.**TokenManager**(*client*)

Bases: *Manager*

authenticate(*username=None*, *tenant_id=None*, *tenant_name=None*, *password=None*, *token=None*, *return_raw=False*)

delete(*token*)

endpoints(*token*)

get_revoked()

Return the revoked tokens response.

The response will be a dict containing signed which is a CMS-encoded document.

get_token_data(*token*)

Fetch the data about a token from the identity server.

Parameters
token (*str*) The token id.

Return type
dict

resource_class

alias of *Token*

validate(*token*)

Validate a token.

Parameters
token Token to be validated.

Return type
Token

validate_access_info(*token*)

Validate a token.

Parameters
token Token to be validated. This can be an instance of *keystoneclient.access.AccessInfo* or a string token_id.

Return type
keystoneclient.access.AccessInfoV2

keystoneclient.v2_0.users module

class keystoneclient.v2_0.users.**User**(*manager*, *info*, *loaded=False*)

Bases: *Resource*

Represents a Keystone user.

delete()

list_roles(*tenant=None*)

class keystoneclient.v2_0.users.**UserManager**(*client*, *role_manager*)

Bases: *ManagerWithFind*

Manager class for manipulating Keystone users.

create(*name*, *password=None*, *email=None*, *tenant_id=None*, *enabled=True*)

Create a user.

delete(*user*)

Delete a user.

get(*user*)

list(*tenant_id=None*, *limit=None*, *marker=None*)

Get a list of users (optionally limited to a tenant).

Return type
list of *User*

list_roles(*user*, *tenant=None*)

resource_class

alias of *User*

update(*user*, ***kwargs*)

Update user data.

Supported arguments include `name`, `email`, and `enabled`.

update_enabled(*user*, *enabled*)

Update enabled-ness.

update_own_password(*origpasswd*, *passwd*)

Update password.

update_password(*user*, *password*)

Update password.

update_tenant(*user*, *tenant*)

Update default tenant.

Module contents

keystoneclient.v3 package

Subpackages

keystoneclient.v3.contrib package

Subpackages

keystoneclient.v3.contrib.federation package

Submodules

keystoneclient.v3.contrib.federation.base module

class keystoneclient.v3.contrib.federation.base.**EntityManager**(*client*)

Bases: *Manager*

Manager class for listing federated accessible objects.

list()

abstract property object_type

resource_class = None

keystoneclient.v3.contrib.federation.core module

class keystoneclient.v3.contrib.federation.core.**FederationManager**(*api*)

Bases: object

keystoneclient.v3.contrib.federation.domains module

class keystoneclient.v3.contrib.federation.domains.**DomainManager**(*client*)

Bases: *EntityManager*

object_type = 'domains'

resource_class

alias of *Domain*

keystoneclient.v3.contrib.federation.identity_providers module

class keystoneclient.v3.contrib.federation.identity_providers.**IdentityProvider**(*manager*, *info*, *loaded=False*)

Bases: *Resource*

Object representing Identity Provider container.

Attributes:

- id: user-defined unique string identifying Identity Provider.

class keystoneclient.v3.contrib.federation.identity_providers.**IdentityProviderManager**(*client*)

Bases: *CrudManager*

Manager class for manipulating Identity Providers.

base_url = 'OS-FEDERATION'

collection_key = 'identity_providers'

create(*id*, ***kwargs*)

Create Identity Provider object.

Utilize Keystone URI: PUT /OS-FEDERATION/identity_providers/$identity_provider

Parameters

- **id** unique id of the identity provider.
- **kwargs** optional attributes: description (str), domain_id (str), enabled (boolean) and remote_ids (list).

Returns
an IdentityProvider resource object.

Return type
keystoneclient.v3.federation.IdentityProvider

delete(*identity_provider*)

Delete Identity Provider object.

Utilize Keystone URI: DELETE /OS-FEDERATION/identity_providers/$identity_provider

Parameters
identity_provider the Identity Provider ID itself or an object with it stored inside.

get(*identity_provider*)

Fetch Identity Provider object.

Utilize Keystone URI: GET /OS-FEDERATION/identity_providers/$identity_provider

Parameters
identity_provider an object with identity_provider_id stored inside.

Returns
an IdentityProvider resource object.

Return type
keystoneclient.v3.federation.IdentityProvider

key = 'identity_provider'

list(***kwargs*)

List all Identity Providers.

Utilize Keystone URI: GET /OS-FEDERATION/identity_providers

Returns
a list of IdentityProvider resource objects.

Return type
List

resource_class

alias of *IdentityProvider*

update(*identity_provider*, ***kwargs*)

Update Identity Provider object.

Utilize Keystone URI: PATCH /OS-FEDERATION/identity_providers/$identity_provider

Parameters
identity_provider an object with identity_provider_id stored inside.

Returns
an IdentityProvider resource object.

Return type
keystoneclient.v3.federation.IdentityProvider

keystoneclient.v3.contrib.federation.mappings module

class keystoneclient.v3.contrib.federation.mappings.**Mapping**(*manager*, *info*, *loaded=False*)

Bases: *Resource*

An object representing mapping container.

Attributes:

- id: user defined unique string identifying mapping.

class keystoneclient.v3.contrib.federation.mappings.**MappingManager**(*client*)

Bases: *CrudManager*

Manager class for manipulating federation mappings.

base_url = 'OS-FEDERATION'

collection_key = 'mappings'

create(*mapping_id*, ***kwargs*)

Create federation mapping.

Utilize Identity API operation: PUT /OS-FEDERATION/mappings/$mapping_id

Parameters

- **mapping_id** user defined string identifier of the federation mapping.
- **rules** a list of mapping rules.

Example of the rules parameter:

```
[
    {
        "local": [
            {
                "group": {
                    "id": "0cd5e9"
                }
            }
        ],
        "remote": [
            {
                "type": "orgPersonType",
                "not_any_of": [
```

(continues on next page)

(continued from previous page)

```
                    "Contractor",
                    "Guest"
                ]
            }
        ]
    }
]
```

delete(*mapping*)

Delete federation mapping identified by mapping id.

Utilize Identity API operation: DELETE /OS-FEDERATION/mappings/$mapping_id

Parameters

mapping a Mapping type object with mapping id stored inside.

get(*mapping*)

Fetch federation mapping identified by mapping id.

Utilize Identity API operation: GET /OS-FEDERATION/mappings/$mapping_id

Parameters

mapping a Mapping type object with mapping id stored inside.

key = 'mapping'

list(***kwargs*)

List all federation mappings.

Utilize Identity API operation: GET /OS-FEDERATION/mappings

resource_class

alias of *Mapping*

update(*mapping*, ***kwargs*)

Update federation mapping identified by mapping id.

Utilize Identity API operation: PATCH /OS-FEDERATION/mappings/$mapping_id

Parameters

- **mapping** a Mapping type object with mapping id stored inside.
- **rules** a list of mapping rules.

Example of the `rules` parameter:

```
[
    {
        "local": [
            {
                "group": {
                    "id": "0cd5e9"
                }
            }
        ],
```

(continues on next page)

(continued from previous page)

```
        "remote": [
            {
                "type": "orgPersonType",
                "not_any_of": [
                    "Contractor",
                    "Guest"
                ]
            }
        ]
    }
]
```

keystoneclient.v3.contrib.federation.projects module

class keystoneclient.v3.contrib.federation.projects.**ProjectManager**(*client*)

Bases: *EntityManager*

object_type = 'projects'

resource_class

alias of *Project*

keystoneclient.v3.contrib.federation.protocols module

class keystoneclient.v3.contrib.federation.protocols.**Protocol**(*manager*, *info*, *loaded=False*)

Bases: *Resource*

An object representing federation protocol container.

Attributes:

- **id: user-defined unique per Identity Provider string identifying** federation protocol.

class keystoneclient.v3.contrib.federation.protocols.**ProtocolManager**(*client*)

Bases: *CrudManager*

Manager class for manipulating federation protocols.

base_url = 'OS-FEDERATION/identity_providers'

build_url(*dict_args_in_out=None*)

Build URL for federation protocols.

collection_key = 'protocols'

create(*protocol_id*, *identity_provider*, *mapping*, ***kwargs*)

Create federation protocol object and tie to the Identity Provider.

Utilize Identity API operation: PUT /OS-FEDERATION/identity_providers/ $identity_provider/protocols/$protocol

Parameters

- **protocol_id** a string type parameter identifying a federation protocol
- **identity_provider** a string type parameter identifying an Identity Provider
- **mapping** a base.Resource object with federation mapping id

delete(*identity_provider*, *protocol*)

Delete Protocol object tied to the Identity Provider.

Utilize Identity API operation: DELETE /OS-FEDERATION/identity_providers/ $identity_provider/protocols/$protocol

Parameters

- **identity_provider** a base.Resource type object with Identity Provider id stored inside
- **protocol** a base.Resource type object with federation protocol id stored inside

get(*identity_provider*, *protocol*, ***kwargs*)

Fetch federation protocol object tied to the Identity Provider.

Utilize Identity API operation: GET /OS-FEDERATION/identity_providers/ $identity_provider/protocols/$protocol

Parameters

- **identity_provider** a base.Resource type object with Identity Provider id stored inside
- **protocol** a base.Resource type object with federation protocol id stored inside

key = 'protocol'

list(*identity_provider*, ***kwargs*)

List all federation protocol objects tied to the Identity Provider.

Utilize Identity API operation: GET /OS-FEDERATION/identity_providers/ $identity_provider/protocols

Parameters

identity_provider a base.Resource type object with Identity Provider id stored inside

resource_class

alias of *Protocol*

update(*identity_provider*, *protocol*, *mapping*, ***kwargs*)

Update Protocol object tied to the Identity Provider.

Utilize Identity API operation: PATCH /OS-FEDERATION/identity_providers/ $identity_provider/protocols/$protocol

Parameters

- **identity_provider** a base.Resource type object with Identity Provider id stored inside
- **protocol** a base.Resource type object with federation protocol id stored inside
- **mapping** a base.Resource object with federation mapping id

keystoneclient.v3.contrib.federation.saml module

class keystoneclient.v3.contrib.federation.saml.**SamlManager**(*client*)

Bases: *Manager*

Manager class for creating SAML assertions.

create_ecp_assertion(*service_provider*, *token_id*)

Create an ECP wrapped SAML assertion from a token.

Equivalent Identity API call: POST /auth/OS-FEDERATION/saml2/ecp

Parameters

- **service_provider** (*string*) Service Provider resource.
- **token_id** (*string*) Token to transform to SAML assertion.

Returns
SAML representation of token_id, wrapped in ECP envelope

Return type
string

create_saml_assertion(*service_provider*, *token_id*)

Create a SAML assertion from a token.

Equivalent Identity API call: POST /auth/OS-FEDERATION/saml2

Parameters

- **service_provider** (*string*) Service Provider resource.
- **token_id** (*string*) Token to transform to SAML assertion.

Returns
SAML representation of token_id

Return type
string

keystoneclient.v3.contrib.federation.service_providers module

class keystoneclient.v3.contrib.federation.service_providers.**ServiceProvider**(*manager*, *info*, *loaded=False*)

Bases: *Resource*

Object representing Service Provider container.

Attributes:

- id: user-defined unique string identifying Service Provider.
- sp_url: the shibboleth endpoint of a Service Provider.
- auth_url: the authentication url of Service Provider.

class keystoneclient.v3.contrib.federation.service_providers.**ServiceProviderManager**(*client*)

Bases: *CrudManager*

Manager class for manipulating Service Providers.

base_url = 'OS-FEDERATION'

collection_key = 'service_providers'

create(*id*, ***kwargs*)

Create Service Provider object.

Utilize Keystone URI: `PUT /OS-FEDERATION/service_providers/{id}`

Parameters
id unique id of the service provider.

delete(*service_provider*)

Delete Service Provider object.

Utilize Keystone URI: `DELETE /OS-FEDERATION/service_providers/{id}`

Parameters
service_provider an object with service_provider_id stored inside.

get(*service_provider*)

Fetch Service Provider object.

Utilize Keystone URI: `GET /OS-FEDERATION/service_providers/{id}`

Parameters
service_provider an object with service_provider_id stored inside.

key = 'service_provider'

list(***kwargs*)

List all Service Providers.

Utilize Keystone URI: `GET /OS-FEDERATION/service_providers`

resource_class

alias of *ServiceProvider*

update(*service_provider*, ***kwargs*)

Update the existing Service Provider object on the server.

Only properties provided to the function are being updated.

Utilize Keystone URI: `PATCH /OS-FEDERATION/service_providers/{id}`

Parameters

service_provider an object with service_provider_id stored inside.

Module contents

keystoneclient.v3.contrib.oauth1 package

Submodules

keystoneclient.v3.contrib.oauth1.access_tokens module

class keystoneclient.v3.contrib.oauth1.access_tokens.**AccessToken**(*manager*, *info*, *loaded=False*)

Bases: *Resource*

class keystoneclient.v3.contrib.oauth1.access_tokens.**AccessTokenManager**(*client*)

Bases: *CrudManager*

Manager class for manipulating identity OAuth access tokens.

create(*consumer_key*, *consumer_secret*, *request_key*, *request_secret*, *verifier*)

resource_class

alias of *AccessToken*

keystoneclient.v3.contrib.oauth1.auth module

class keystoneclient.v3.contrib.oauth1.auth.**OAuth**(*auth_url*, **args*, ***kwargs*)

Bases: *AuthConstructor*

class keystoneclient.v3.contrib.oauth1.auth.**OAuthMethod**(***kwargs*)

Bases: *AuthMethod*

OAuth based authentication method.

Parameters

- **consumer_key** (*string*) Consumer key.
- **consumer_secret** (*string*) Consumer secret.
- **access_key** (*string*) Access token key.
- **access_secret** (*string*) Access token secret.

get_auth_data(*session*, *auth*, *headers*, ***kwargs*)

Return the authentication section of an auth plugin.

Parameters

- **session** (keystoneclient.session.Session) The communication session.
- **auth** (base.Auth) The auth plugin calling the method.
- **headers** (*dict*) The headers that will be sent with the auth request if a plugin needs to add to them.

Returns

The identifier of this plugin and a dict of authentication data for the auth type.

Return type

tuple(string, dict)

keystoneclient.v3.contrib.oauth1.consumers module

class keystoneclient.v3.contrib.oauth1.consumers.**Consumer**(*manager*, *info*, *loaded=False*)

Bases: *Resource*

Represents an OAuth consumer.

Attributes:

- id: a uuid that identifies the consumer
- description: a short description of the consumer

class keystoneclient.v3.contrib.oauth1.consumers.**ConsumerManager**(*client*)

Bases: *CrudManager*

Manager class for manipulating identity consumers.

base_url = '/OS-OAUTH1'

collection_key = 'consumers'

create(*description=None*, ***kwargs*)

delete(*consumer*)

get(*consumer*)

key = 'consumer'

resource_class

alias of *Consumer*

update(*consumer*, *description=None*, ***kwargs*)

keystoneclient.v3.contrib.oauth1.core module

class keystoneclient.v3.contrib.oauth1.core.**OAuthManager**(*api*)

Bases: object

class keystoneclient.v3.contrib.oauth1.core.**OAuthManagerOptionalImportProxy**

Bases: object

Act as a proxy manager in case oauthlib is no installed.

This class will only be created if oauthlib is not in the system, trying to access any of the attributes in name (access_tokens, consumers, request_tokens), will result in a NotImplementedError, and a message.

```
>>> manager.access_tokens.blah
NotImplementedError: To use 'access_tokens' oauthlib must be installed
```

Otherwise, if trying to access an attribute other than the ones in name, the manager will state that the attribute does not exist.

```
>>> manager.dne.blah
AttributeError: 'OAuthManagerOptionalImportProxy' object has no
attribute 'dne'
```

keystoneclient.v3.contrib.oauth1.core.**create_oauth_manager**(*self*)

keystoneclient.v3.contrib.oauth1.request_tokens module

class keystoneclient.v3.contrib.oauth1.request_tokens.**RequestToken**(*manager*, *info*, *loaded=False*)

Bases: *Resource*

authorize(*roles*)

class keystoneclient.v3.contrib.oauth1.request_tokens.**RequestTokenManager**(*client*)

Bases: *CrudManager*

Manager class for manipulating identity OAuth request tokens.

authorize(*request_token*, *roles*)

Authorize a request token with specific roles.

Utilize Identity API operation: PUT /OS-OAUTH1/authorize/$request_token_id

Parameters

- **request_token** a request token that will be authorized, and can be exchanged for an access token.
- **roles** a list of roles, that will be delegated to the user.

create(*consumer_key*, *consumer_secret*, *project*)

resource_class

alias of *RequestToken*

keystoneclient.v3.contrib.oauth1.utils module

keystoneclient.v3.contrib.oauth1.utils.**get_oauth_token_from_body**(*body*)

Parse the URL response body to retrieve the oauth token key and secret.

The response body will look like: oauth_token=12345&oauth_token_secret=67890 with oauth_expires_at=2013-03-30T05:27:19.463201 possibly there, too.

Module contents

Submodules

keystoneclient.v3.contrib.endpoint_filter module

class keystoneclient.v3.contrib.endpoint_filter.**EndpointFilterManager**(*client*)

Bases: *Manager*

Manager class for manipulating project-endpoint associations.

Project-endpoint associations can be with endpoints directly or via endpoint groups.

OS_EP_FILTER_EXT = '/OS-EP-FILTER'

add_endpoint_group_to_project(*endpoint_group*, *project*)

Create a project-endpoint group association.

add_endpoint_to_project(*project*, *endpoint*)

Create a project-endpoint association.

check_endpoint_group_in_project(*endpoint_group*, *project*)

Check if project-endpoint group association exists.

check_endpoint_in_project(*project*, *endpoint*)

Check if project-endpoint association exists.

delete_endpoint_from_project(*project*, *endpoint*)

Remove a project-endpoint association.

delete_endpoint_group_from_project(*endpoint_group*, *project*)

Remove a project-endpoint group association.

list_endpoint_groups_for_project(*project*)

List all endpoint groups for a given project.

list_endpoints_for_project(*project*)

List all endpoints for a given project.

list_projects_for_endpoint(*endpoint*)

List all projects for a given endpoint.

list_projects_for_endpoint_group(*endpoint_group*)

List all projects associated with a given endpoint group.

keystoneclient.v3.contrib.endpoint_policy module

class keystoneclient.v3.contrib.endpoint_policy.**EndpointPolicyManager**(*client*)

Bases: *Manager*

Manager class for manipulating endpoint-policy associations.

OS_EP_POLICY_EXT = 'OS-ENDPOINT-POLICY'

check_policy_association_for_endpoint(*policy*, *endpoint*)

Check an association between a policy and an endpoint.

check_policy_association_for_region_and_service(*policy*, *region*, *service*)

Check an association between a policy and a service in a region.

check_policy_association_for_service(*policy*, *service*)

Check an association between a policy and a service.

create_policy_association_for_endpoint(*policy*, *endpoint*)

Create an association between a policy and an endpoint.

create_policy_association_for_region_and_service(*policy*, *region*, *service*)

Create an association between a policy and a service in a region.

create_policy_association_for_service(*policy*, *service*)

Create an association between a policy and a service.

delete_policy_association_for_endpoint(*policy*, *endpoint*)

Delete an association between a policy and an endpoint.

delete_policy_association_for_region_and_service(*policy*, *region*, *service*)

Delete an association between a policy and a service in a region.

delete_policy_association_for_service(*policy*, *service*)

Delete an association between a policy and a service.

get_policy_for_endpoint(*endpoint*)

Get the effective policy for an endpoint.

Parameters
endpoint endpoint object or ID

Returns
policies.Policy object

list_endpoints_for_policy(*policy*)

List endpoints with the effective association to a policy.

Parameters
policy policy object or ID

Returns
list of endpoints that are associated with the policy

keystoneclient.v3.contrib.simple_cert module

class keystoneclient.v3.contrib.simple_cert.**SimpleCertManager**(*client*)

Bases: object

Manager for the OS-SIMPLE-CERT extension.

get_ca_certificates()

Get CA certificates.

Returns
PEM-formatted string.

Return type
str

get_certificates()

Get signing certificates.

Returns
PEM-formatted string.

Return type
str

keystoneclient.v3.contrib.trusts module

class keystoneclient.v3.contrib.trusts.**Trust**(*manager*, *info*, *loaded=False*)

Bases: *Resource*

Represents a Trust.

Attributes:

- id: a uuid that identifies the trust
- impersonation: allow explicit impersonation
- project_id: project ID
- trustee_user_id: a uuid that identifies the trustee
- trustor_user_id: a uuid that identifies the trustor

class keystoneclient.v3.contrib.trusts.**TrustManager**(*client*)

Bases: *CrudManager*

Manager class for manipulating Trusts.

base_url = '/OS-TRUST'

collection_key = 'trusts'

create(*trustee_user*, *trustor_user*, *role_names=None*, *role_ids=None*, *project=None*, *impersonation=False*, *expires_at=None*, *remaining_uses=None*, ***kwargs*)

Create a Trust.

Parameters

- **trustee_user** (*string*) user who is capable of consuming the trust
- **trustor_user** (*string*) user whos authorization is being delegated
- **role_names** (*string*) subset of trustors roles to be granted
- **role_ids** (*string*) subset of trustors roles to be granted
- **project** (*string*) project which the trustor is delegating
- **impersonation** (*boolean*) enable explicit impersonation
- **expires_at** (*datetime.datetime*) expiry time
- **remaining_uses** (*integer*) how many times this trust can be used to generate a token. None means unlimited tokens.

delete(*trust*)

Delete a trust.

get(*trust*)

Get a specific trust.

key = 'trust'

list(*trustee_user=None*, *trustor_user=None*, ***kwargs*)

List Trusts.

resource_class

alias of *Trust*

update()

Module contents

Submodules

keystoneclient.v3.access_rules module

class keystoneclient.v3.access_rules.**AccessRule**(*manager*, *info*, *loaded=False*)

Bases: *Resource*

Represents an Identity access rule for application credentials.

Attributes:

- id: a uuid that identifies the access rule
- **method: The request method that the application credential is** permitted to use for a given API endpoint
- **path: The API path that the application credential is permitted to** access
- **service: The service type identifier for the service that the** application credential is permitted to access

class keystoneclient.v3.access_rules.**AccessRuleManager**(*client*)

Bases: *CrudManager*

Manager class for manipulating Identity access rules.

collection_key = 'access_rules'

create()

delete(*access_rule*, *user=None*)

Delete an access rule.

Parameters

- **access_rule** (str or *keystoneclient.v3.access_rules.AccessRule*) the access rule to be deleted
- **user** (*string*) User ID

Returns

response object with 204 status

Return type

requests.models.Response

find(*user=None*, ***kwargs*)

Find an access rule with attributes matching **kwargs.

Parameters

user (*string*) User ID

Returns

a list of matching access rules

Return type

list of *keystoneclient.v3.access_rules.AccessRule*

get(*access_rule*, *user=None*)

Retrieve an access rule.

Parameters

- **access_rule** (str or *keystoneclient.v3.access_rules.AccessRule*) the access rule to be retrieved from the server
- **user** (*string*) User ID

Returns

the specified access rule

Return type

keystoneclient.v3.access_rules.AccessRule

key = 'access_rule'

list(*user=None*, ***kwargs*)

List access rules.

Parameters

user (*string*) User ID

Returns
a list of access rules

Return type
list of *keystoneclient.v3.access_rules.AccessRule*

resource_class

alias of *AccessRule*

update()

keystoneclient.v3.application_credentials module

class keystoneclient.v3.application_credentials.**ApplicationCredential**(*manager*, *info*, *loaded=False*)

Bases: *Resource*

Represents an Identity application credential.

Attributes:

- id: a uuid that identifies the application credential
- user: the user who owns the application credential
- name: application credential name
- secret: application credential secret
- description: application credential description
- expires_at: expiry time
- roles: role assignments on the project
- **unrestricted: whether the application credential has restrictions** applied
- **access_rules: a list of access rules defining what API requests the** application credential may be used for

class keystoneclient.v3.application_credentials.**ApplicationCredentialManager**(*client*)

Bases: *CrudManager*

Manager class for manipulating Identity application credentials.

collection_key = 'application_credentials'

create(*name*, *user=None*, *secret=None*, *description=None*, *expires_at=None*, *roles=None*, *unrestricted=False*, *access_rules=None*, ***kwargs*)

Create a credential.

Parameters

- **name** (*string*) application credential name
- **user** (*string*) User ID
- **secret** application credential secret

- **description** application credential description
- **expires_at** (*datetime.datetime*) expiry time
- **roles** (*List*) list of roles on the project. Maybe a list of IDs or a list of dicts specifying role name and domain
- **unrestricted** (*bool*) whether the application credential has restrictions applied
- **access_rules** (*List*) a list of dicts representing access rules

Returns
the created application credential

Return type
keystoneclient.v3.application_credentials.ApplicationCredential

delete(*application_credential*, *user=None*)

Delete an application credential.

Parameters
application_credential the application credential to be deleted

Returns
response object with 204 status

Return type
requests.models.Response

find(*user=None*, ***kwargs*)

Find an application credential with attributes matching **kwargs.

Parameters
user (*string*) User ID

Returns
a list of matching application credentials

Return type
list of *keystoneclient.v3.application_credentials.ApplicationCredential*

get(*application_credential*, *user=None*)

Retrieve an application credential.

Parameters
application_credential the credential to be retrieved from the server

Returns
the specified application credential

Return type
keystoneclient.v3.application_credentials.ApplicationCredential

key = 'application_credential'

list(*user=None*, ***kwargs*)

List application credentials.

Parameters
user (*string*) User ID

Returns
a list of application credentials

Return type
list of *keystoneclient.v3.application_credentials.ApplicationCredential*

resource_class

alias of *ApplicationCredential*

update()

keystoneclient.v3.auth module

class keystoneclient.v3.auth.**AuthManager**(*client*)

Bases: *Manager*

Retrieve auth context specific information.

The information returned by the auth routes is entirely dependent on the authentication information provided by the user.

domains()

List Domains that the specified token can be rescoped to.

Returns
a list of domains.

Return type
list of *keystoneclient.v3.domains.Domain*.

projects()

List projects that the specified token can be rescoped to.

Returns
a list of projects.

Return type
list of *keystoneclient.v3.projects.Project*

systems()

List Systems that the specified token can be rescoped to.

At the moment this is either empty or all.

Returns
a list of systems.

Return type
list of keystoneclient.v3.systems.System.

keystoneclient.v3.client module

class keystoneclient.v3.client.**Client**(***kwargs*)

Bases: *HTTPClient*

Client for the OpenStack Identity API v3.

Parameters

- **session** (*keystoneauth1.session.Session*) Session for requests. (optional)
- **user_id** (*string*) User ID for authentication. (optional)
- **username** (*string*) Username for authentication. (optional)
- **user_domain_id** (*string*) Users domain ID for authentication. (optional)
- **user_domain_name** (*string*) Users domain name for authentication. (optional)
- **password** (*string*) Password for authentication. (optional)
- **token** (*string*) Token for authentication. (optional)
- **domain_id** (*string*) Domain ID for domain scoping. (optional)
- **domain_name** (*string*) Domain name for domain scoping. (optional)
- **project_id** (*string*) Project ID for project scoping. (optional)
- **project_name** (*string*) Project name for project scoping. (optional)
- **project_domain_id** (*string*) Projects domain ID for project scoping. (optional)
- **project_domain_name** (*string*) Projects domain name for project scoping. (optional)
- **tenant_name** (*string*) Tenant name. (optional) The tenant_name keyword argument is deprecated as of the 1.7.0 release in favor of project_name and may be removed in the 2.0.0 release.
- **tenant_id** (*string*) Tenant id. (optional) The tenant_id keyword argument is deprecated as of the 1.7.0 release in favor of project_id and may be removed in the 2.0.0 release.
- **auth_url** (*string*) Identity service endpoint for authorization.
- **region_name** (*string*) Name of a region to select when choosing an endpoint from the service catalog.
- **endpoint** (*string*) A user-supplied endpoint URL for the identity service. Lazy-authentication is possible for API service calls if endpoint is set at instantiation. (optional)
- **timeout** (*integer*) Allows customization of the timeout for client http requests. (optional)

Warning: Constructing an instance of this class without a session is deprecated as of the 1.7.0 release and will be removed in the 2.0.0 release.

Example:

```
>>> from keystoneauth1.identity import v3
>>> from keystoneauth1 import session
>>> from keystoneclient.v3 import client
>>> auth = v3.Password(user_domain_name=DOMAIN_NAME,
...                    username=USER,
...                    password=PASS,
...                    project_domain_name=PROJECT_DOMAIN_NAME,
...                    project_name=PROJECT_NAME,
...                    auth_url=KEYSTONE_URL)
>>> sess = session.Session(auth=auth)
>>> keystone = client.Client(session=sess)
>>> keystone.projects.list()
...
>>> user = keystone.users.get(USER_ID)
>>> user.delete()
```

Instances of this class have the following managers:

credentials

keystoneclient.v3.credentials.CredentialManager

domain_configs

keystoneclient.v3.domain_configs.DomainConfigManager

ec2

keystoneclient.v3.ec2.EC2Manager

endpoint_filter

keystoneclient.v3.contrib.endpoint_filter.EndpointFilterManager

endpoint_groups

keystoneclient.v3.endpoint_groups.EndpointGroupManager

endpoint_policy

keystoneclient.v3.contrib.endpoint_policy.EndpointPolicyManager

endpoints

keystoneclient.v3.endpoints.EndpointManager

domains

keystoneclient.v3.domains.DomainManager

federation

keystoneclient.v3.contrib.federation.core.FederationManager

groups

keystoneclient.v3.groups.GroupManager

limits

keystoneclient.v3.limits.LimitManager

oauth1

keystoneclient.v3.contrib.oauth1.core.OAuthManager

policies

keystoneclient.v3.policies.PolicyManager

regions

keystoneclient.v3.regions.RegionManager

registered_limits

keystoneclient.v3.registered_limits.RegisteredLimitManager

role_assignments

keystoneclient.v3.role_assignments.RoleAssignmentManager

roles

keystoneclient.v3.roles.RoleManager

simple_cert

keystoneclient.v3.contrib.simple_cert.SimpleCertManager

services

keystoneclient.v3.services.ServiceManager

tokens

keystoneclient.v3.tokens.TokenManager

trusts

keystoneclient.v3.contrib.trusts.TrustManager

users

keystoneclient.v3.users.UserManager

get_raw_token_from_identity_service(*auth_url*, *user_id=None*, *username=None*, *user_domain_id=None*, *user_domain_name=None*, *password=None*, *domain_id=None*, *domain_name=None*, *project_id=None*, *project_name=None*, *project_domain_id=None*, *project_domain_name=None*, *token=None*, *trust_id=None*, ***kwargs*)

Authenticate against the v3 Identity API.

If password and token methods are both provided then both methods will be used in the request.

Returns

access.AccessInfo if authentication was successful.

Return type

keystoneclient.access.AccessInfoV3

Raises

- *keystoneclient.exceptions.AuthorizationFailure* if unable to authenticate or validate the existing authorization token.
- *keystoneclient.exceptions.Unauthorized* if authentication fails due to invalid token.

process_token(***kwargs*)

Extract and process information from the new auth_ref.

And set the relevant authentication information.

serialize(*entity*)

version = 'v3'

keystoneclient.v3.credentials module

class keystoneclient.v3.credentials.**Credential**(*manager*, *info*, *loaded=False*)

Bases: *Resource*

Represents an Identity credential.

Attributes:

- id: a uuid that identifies the credential
- user_id: user ID to which credential belongs
- type: the type of credential
- blob: the text that represents the credential
- project_id: project ID which limits the scope of the credential

class keystoneclient.v3.credentials.**CredentialManager**(*client*)

Bases: *CrudManager*

Manager class for manipulating Identity credentials.

collection_key = 'credentials'

create(*user*, *type*, *blob*, *project=None*, ***kwargs*)

Create a credential.

Parameters

- **user** (str or *keystoneclient.v3.users.User*) the user to which the credential belongs
- **type** (*str*) the type of the credential, valid values are: `ec2`, `cert` or `totp`
- **blob** (*str*) the arbitrary blob of the credential data, to be parsed according to the type
- **project** (str or *keystoneclient.v3.projects.Project*) the project which limits the scope of the credential, this attribbute is mandatory if the credential type is ec2
- **kwargs** any other attribute provided will be passed to the server

Returns
the created credential

Return type
keystoneclient.v3.credentials.Credential

delete(*credential*)

Delete a credential.

Parameters
credential (str or *keystoneclient.v3.credentials.Credential*) the credential to be deleted

Returns
response object with 204 status

Return type
requests.models.Response

get(*credential*)

Retrieve a credential.

Parameters
credential (str or *keystoneclient.v3.credentials.Credential*) the credential to be retrieved from the server

Returns
the specified credential

Return type
keystoneclient.v3.credentials.Credential

key = 'credential'

list(***kwargs*)

List credentials.

Parameters
kwargs If user_id or type is specified then credentials will be filtered accordingly.

Returns
a list of credentials

Return type
list of *keystoneclient.v3.credentials.Credential*

resource_class

alias of *Credential*

update(*credential*, *user*, *type=None*, *blob=None*, *project=None*, ***kwargs*)

Update a credential.

Parameters

- **credential** (str or *keystoneclient.v3.credentials.Credential*) the credential to be updated on the server
- **user** (str or *keystoneclient.v3.users.User*) the new user to which the credential belongs

- **type** (*str*) the new type of the credential, valid values are: `ec2`, `cert` or `totp`
- **blob** (*str*) the new blob of the credential data and may be removed in the future release.
- **project** (str or *keystoneclient.v3.projects.Project*) the new project which limits the scope of the credential, this attribute is mandatory if the credential type is ec2
- **kwargs** any other attribute provided will be passed to the server

Returns
the updated credential

Return type
keystoneclient.v3.credentials.Credential

keystoneclient.v3.domain_configs module

class keystoneclient.v3.domain_configs.**DomainConfig**(*manager*, *info*, *loaded=False*)

Bases: *Resource*

An object representing a domain config association.

This resource object does not necessarily contain fixed attributes, as new attributes are added in the server, they are supported here directly. The currently supported configs are *identity* and *ldap*.

class keystoneclient.v3.domain_configs.**DomainConfigManager**(*client*)

Bases: *Manager*

Manager class for manipulating domain config associations.

build_url(*domain*)

create(*domain*, *config*)

Create a config for a domain.

Parameters

- **domain** (str or *keystoneclient.v3.domains.Domain*) the domain where the config is going to be applied.
- **config** (*dict*) a dictionary of domain configurations.

Example of the `config` parameter:

```
{
    "identity": {
        "driver": "ldap"
    },
    "ldap": {
        "url": "ldap://myldap.com:389/",
        "user_tree_dn": "ou=Users,dc=my_new_root,dc=org"
    }
}
```

Returns
the created domain config returned from server.

Return type
keystoneclient.v3.domain_configs.DomainConfig

delete(*domain*)

Delete a config for a domain.

Parameters
domain (str or *keystoneclient.v3.domains.Domain*) the domain which the config will be deleted on the server.

Returns
Response object with 204 status.

Return type
requests.models.Response

find(***kwargs*)

get(*domain*)

Get a config for a domain.

Parameters
domain (str or *keystoneclient.v3.domains.Domain*) the domain for which the config is defined.

Returns
the domain config returned from server.

Return type
keystoneclient.v3.domain_configs.DomainConfig

key = 'config'

list(***kwargs*)

resource_class

alias of *DomainConfig*

update(*domain*, *config*)

Update a config for a domain.

Parameters

- **domain** (str or *keystoneclient.v3.domains.Domain*) the domain where the config is going to be updated.
- **config** (*dict*) a dictionary of domain configurations.

Example of the config parameter:

```
{
    "identity": {
        "driver": "ldap"
    },
    "ldap": {
```

(continues on next page)

(continued from previous page)

```
        "url": "ldap://myldap.com:389/",
        "user_tree_dn": "ou=Users,dc=my_new_root,dc=org"
    }
}
```

Returns
the updated domain config returned from server.

Return type
keystoneclient.v3.domain_configs.DomainConfig

keystoneclient.v3.domains module

class keystoneclient.v3.domains.**Domain**(*manager*, *info*, *loaded=False*)

Bases: *Resource*

Represents an Identity domain.

Attributes:

- id: a uuid that identifies the domain
- name: the name of the domain
- description: a description of the domain
- enabled: determines whether the domain is enabled

class keystoneclient.v3.domains.**DomainManager**(*client*)

Bases: *CrudManager*

Manager class for manipulating Identity domains.

collection_key = 'domains'

create(*name*, *description=None*, *enabled=True*, ***kwargs*)

Create a domain.

Parameters

- **name** (*str*) the name of the domain.
- **description** (*str*) a description of the domain.
- **enabled** (*bool*) whether the domain is enabled.
- **kwargs** any other attribute provided will be passed to the server.

Returns
the created domain returned from server.

Return type
keystoneclient.v3.domains.Domain

delete(*domain*)

Delete a domain.

Parameters
domain (str or *keystoneclient.v3.domains.Domain*) the domain to be deleted on the server.

Returns
Response object with 204 status.

Return type
requests.models.Response

get(*domain*)

Retrieve a domain.

Parameters
domain (str or *keystoneclient.v3.domains.Domain*) the domain to be retrieved from the server.

Returns
the specified domain returned from server.

Return type
keystoneclient.v3.domains.Domain

key = 'domain'

list(***kwargs*)

List domains.

Parameters
kwargs allows filter criteria to be passed where supported by the server.

Returns
a list of domains.

Return type
list of *keystoneclient.v3.domains.Domain*.

resource_class

alias of *Domain*

update(*domain*, *name=None*, *description=None*, *enabled=None*, ***kwargs*)

Update a domain.

Parameters

- **domain** (str or *keystoneclient.v3.domains.Domain*) the domain to be updated on the server.
- **name** (*str*) the new name of the domain.
- **description** (*str*) the new description of the domain.
- **enabled** (*bool*) whether the domain is enabled.
- **kwargs** any other attribute provided will be passed to the server.

Returns
the updated domain returned from server.

Return type
keystoneclient.v3.domains.Domain

keystoneclient.v3.ec2 module

class keystoneclient.v3.ec2.**EC2**(*manager*, *info*, *loaded=False*)

Bases: *Resource*

Represents an EC2 resource.

Attributes:

- id: a string that identifies the EC2 resource.
- user_id: the ID field of a pre-existing user in the backend.
- project_id: the ID field of a pre-existing project in the backend.
- access: a string representing access key of the access/secret pair.
- secret: a string representing the secret of the access/secret pair.

class keystoneclient.v3.ec2.**EC2Manager**(*client*)

Bases: *ManagerWithFind*

create(*user_id*, *project_id*)

Create a new access/secret pair.

Parameters

- **user_id** (str or *keystoneclient.v3.users.User*) the ID of the user having access/secret pair.
- **project_id** (str or *keystoneclient.v3.projects.Project*) the ID of the project having access/secret pair.

Returns

the created access/secret pair returned from server.

Return type

keystoneclient.v3.ec2.EC2

delete(*user_id*, *access*)

Delete an access/secret pair.

Parameters

- **user_id** (str or *keystoneclient.v3.users.User*) the ID of the user whose access/secret pair will be deleted on the server.
- **access** (*str*) the access key whose access/secret pair will be deleted on the server.

Returns

Response object with 204 status.

Return type

requests.models.Response

get(*user_id*, *access*)

Retrieve an access/secret pair for a given access key.

Parameters

- **user_id** (str or *keystoneclient.v3.users.User*) the ID of the user whose access/secret pair will be retrieved from the server.
- **access** (*str*) the access key whose access/secret pair will be retrieved from the server.

Returns
the specified access/secret pair returned from server.

Return type
keystoneclient.v3.ec2.EC2

`list`(*user_id*)

List access/secret pairs for a given user.

Parameters
user_id (*str*) the ID of the user having access/secret pairs will be listed.

Returns
a list of access/secret pairs.

Return type
list of *keystoneclient.v3.ec2.EC2*

`resource_class`

alias of *EC2*

keystoneclient.v3.endpoint_groups module

`class keystoneclient.v3.endpoint_groups.`**EndpointGroup**(*manager*, *info*, *loaded=False*)

Bases: *Resource*

Represents an identity endpoint group.

Attributes:

- id: a UUID that identifies the endpoint group
- name: the endpoint group name
- description: the endpoint group description
- **filters: representation of filters in the format of JSON that define** what endpoint entities are part of the group

`class keystoneclient.v3.endpoint_groups.`**EndpointGroupManager**(*client*)

Bases: *CrudManager*

Manager class for Endpoint Groups.

`base_url = 'OS-EP-FILTER'`

`check`(*endpoint_group*)

Check if an endpoint group exists.

Parameters
endpoint_group (str or *keystoneclient.v3.endpoint_groups.EndpointGroup*) the endpoint group to be checked against the server.

Returns
none if the specified endpoint group exists.

`collection_key = 'endpoint_groups'`

`create`(*name*, *filters*, *description=None*, ***kwargs*)

Create an endpoint group.

Parameters

- **name** (*str*) the name of the endpoint group.
- **filters** (*str*) representation of filters in the format of JSON that define what endpoint entities are part of the group.
- **description** (*str*) a description of the endpoint group.
- **kwargs** any other attribute provided will be passed to the server.

Returns
the created endpoint group returned from server.

Return type
keystoneclient.v3.endpoint_groups.EndpointGroup

`delete`(*endpoint_group*)

Delete an endpoint group.

Parameters
endpoint_group (str or *keystoneclient.v3.endpoint_groups.EndpointGroup*) the endpoint group to be deleted on the server.

Returns
Response object with 204 status.

Return type
`requests.models.Response`

`get`(*endpoint_group*)

Retrieve an endpoint group.

Parameters
endpoint_group (str or *keystoneclient.v3.endpoint_groups.EndpointGroup*) the endpoint group to be retrieved from the server.

Returns
the specified endpoint group returned from server.

Return type
keystoneclient.v3.endpoint_groups.EndpointGroup

`key = 'endpoint_group'`

`list`(***kwargs*)

List endpoint groups.

Any parameter provided will be passed to the server.

Returns
a list of endpoint groups.

Return type
list of *keystoneclient.v3.endpoint_groups.EndpointGroup*.

resource_class
alias of *EndpointGroup*

update(*endpoint_group*, *name=None*, *filters=None*, *description=None*, ***kwargs*)
Update an endpoint group.

Parameters

- **name** (*str*) the new name of the endpoint group.
- **filters** (*str*) the new representation of filters in the format of JSON that define what endpoint entities are part of the group.
- **description** (*str*) the new description of the endpoint group.
- **kwargs** any other attribute provided will be passed to the server.

Returns
the updated endpoint group returned from server.

Return type
keystoneclient.v3.endpoint_groups.EndpointGroup

keystoneclient.v3.endpoints module

class keystoneclient.v3.endpoints.**Endpoint**(*manager*, *info*, *loaded=False*)
Bases: *Resource*

Represents an Identity endpoint.

Attributes:

- id: a uuid that identifies the endpoint
- interface: public, admin or internal network interface
- region: geographic location of the endpoint
- service_id: service to which the endpoint belongs
- url: fully qualified service endpoint
- **enabled: determines whether the endpoint appears in the service** catalog

class keystoneclient.v3.endpoints.**EndpointManager**(*client*)
Bases: *CrudManager*

Manager class for manipulating Identity endpoints.

collection_key = 'endpoints'

create(*service*, *url*, *interface=None*, *region=None*, *enabled=True*, ***kwargs*)
Create an endpoint.

Parameters

- **service** (str or *keystoneclient.v3.services.Service*) the service to which the endpoint belongs.
- **url** (*str*) the URL of the fully qualified service endpoint.
- **interface** (*str*) the network interface of the endpoint. Valid values are: `public`, `admin` or `internal`.
- **region** (str or *keystoneclient.v3.regions.Region*) the region to which the endpoint belongs.
- **enabled** (*bool*) whether the endpoint is enabled or not, determining if it appears in the service catalog.
- **kwargs** any other attribute provided will be passed to the server.

Returns
the created endpoint returned from server.

Return type
keystoneclient.v3.endpoints.Endpoint

delete(*endpoint*)

Delete an endpoint.

Parameters
endpoint (str or *keystoneclient.v3.endpoints.Endpoint*) the endpoint to be deleted on the server.

Returns
Response object with 204 status.

Return type
`requests.models.Response`

get(*endpoint*)

Retrieve an endpoint.

Parameters
endpoint (str or *keystoneclient.v3.endpoints.Endpoint*) the endpoint to be retrieved from the server.

Returns
the specified endpoint returned from server.

Return type
keystoneclient.v3.endpoints.Endpoint

key = 'endpoint'

list(*service=None*, *interface=None*, *region=None*, *enabled=None*, *region_id=None*, ***kwargs*)

List endpoints.

Parameters

- **service** (str or *keystoneclient.v3.services.Service*) the service of the endpoints to be filtered on.

- **interface** (*str*) the network interface of the endpoints to be filtered on. Valid values are: `public`, `admin` or `internal`.
- **enabled** (*bool*) whether to return enabled or disabled endpoints.
- **region_id** (*str*) filter endpoints by the region_id attribute. If both region and region_id are specified, region takes precedence.
- **kwargs** any other attribute provided will filter endpoints on.

Returns
a list of endpoints.

Return type
list of *keystoneclient.v3.endpoints.Endpoint*

resource_class

alias of *Endpoint*

update(*endpoint*, *service=None*, *url=None*, *interface=None*, *region=None*, *enabled=None*, ***kwargs*)

Update an endpoint.

Parameters

- **endpoint** (str or *keystoneclient.v3.endpoints.Endpoint*) the endpoint to be updated on the server.
- **service** (str or *keystoneclient.v3.services.Service*) the new service to which the endpoint belongs.
- **url** (*str*) the new URL of the fully qualified service endpoint.
- **interface** (*str*) the new network interface of the endpoint. Valid values are: `public`, `admin` or `internal`.
- **region** (str or *keystoneclient.v3.regions.Region*) the new region to which the endpoint belongs.
- **enabled** (*bool*) determining if the endpoint appears in the service catalog by enabling or disabling it.
- **kwargs** any other attribute provided will be passed to the server.

Returns
the updated endpoint returned from server.

Return type
keystoneclient.v3.endpoints.Endpoint

keystoneclient.v3.groups module

class keystoneclient.v3.groups.**Group**(*manager*, *info*, *loaded=False*)

Bases: *Resource*

Represents an Identity user group.

Attributes:

- id: a uuid that identifies the group
- name: group name
- description: group description

update(*name=None*, *description=None*)

class keystoneclient.v3.groups.**GroupManager**(*client*)

Bases: *CrudManager*

Manager class for manipulating Identity groups.

collection_key = 'groups'

create(*name*, *domain=None*, *description=None*, ***kwargs*)

Create a group.

Parameters

- **name** (*str*) the name of the group.
- **domain** (str or *keystoneclient.v3.domains.Domain*) the domain of the group.
- **description** (*str*) a description of the group.
- **kwargs** any other attribute provided will be passed to the server.

Returns
the created group returned from server.

Return type
keystoneclient.v3.groups.Group

delete(*group*)

Delete a group.

Parameters
group (str or *keystoneclient.v3.groups.Group*) the group to be deleted on the server.

Returns
Response object with 204 status.

Return type
requests.models.Response

get(*group*)

Retrieve a group.

Parameters
group (str or *keystoneclient.v3.groups.Group*) the group to be retrieved from the server.

Returns
the specified group returned from server.

Return type
keystoneclient.v3.groups.Group

`key = 'group'`

`list`(*user=None*, *domain=None*, ***kwargs*)

List groups.

Parameters

- **user** (str or *keystoneclient.v3.users.User*) the user of the groups to be filtered on.
- **domain** (str or *keystoneclient.v3.domains.Domain*) the domain of the groups to be filtered on.
- **kwargs** any other attribute provided will filter groups on.

Returns
a list of groups.

Return type
list of *keystoneclient.v3.groups.Group*.

`resource_class`

alias of *Group*

`update`(*group*, *name=None*, *description=None*, ***kwargs*)

Update a group.

Parameters

- **group** (str or *keystoneclient.v3.groups.Group*) the group to be updated on the server.
- **name** (*str*) the new name of the group.
- **description** (*str*) the new description of the group.
- **kwargs** any other attribute provided will be passed to server.

Returns
the updated group returned from server.

Return type
keystoneclient.v3.groups.Group

keystoneclient.v3.limits module

class keystoneclient.v3.limits.**Limit**(*manager*, *info*, *loaded=False*)

Bases: *Resource*

Represents a project limit.

Attributes:

- id: a UUID that identifies the project limit
- service_id: a UUID that identifies the service for the limit
- region_id: a UUID that identifies the region for the limit
- project_id: a UUID that identifies the project for the limit
- resource_name: the name of the resource to limit
- resource_limit: the limit to apply to the project
- description: a description for the project limit

class keystoneclient.v3.limits.**LimitManager**(*client*)

Bases: *CrudManager*

Manager class for project limits.

collection_key = 'limits'

create(*project*, *service*, *resource_name*, *resource_limit*, *description=None*, *region=None*, ***kwargs*)

Create a project-specific limit.

Parameters

- **project** (str or *keystoneclient.v3.projects.Project*) the project to create a limit for.
- **service** (str or *keystoneclient.v3.services.Service*) the service that owns the resource to limit.
- **resource_name** (*str*) the name of the resource to limit
- **resource_limit** (*int*) the quantity of the limit
- **description** (*str*) a description of the limit
- **region** (str or *keystoneclient.v3.regions.Region*) region the limit applies to

Returns

a reference of the created limit

Return type

keystoneclient.v3.limits.Limit

delete(*limit*)

Delete a project-specific limit.

Parameters
limit (str or keystoneclient.v3.limit.Limit) the project-specific limit to be deleted.

Returns
Response object with 204 status

Return type
requests.models.Response

get(*limit*)

Retrieve a project limit.

Parameters
limit (str or keystoneclient.v3.limit.Limit) the project-specific limit to be retrieved.

Returns
a project-specific limit

Return type
keystoneclient.v3.limit.Limit

key = 'limit'

list(*service=None*, *region=None*, *resource_name=None*, ***kwargs*)

List project-specific limits.

Any parameter provided will be passed to the server as a filter

Parameters

- **service** (UUID or *keystoneclient.v3.services.Service*) service to filter limits by
- **region** (UUID or *keystoneclient.v3.regions.Region*) region to filter limits by
- **resource_name** (*str*) the name of the resource to filter limits by

Returns
a list of project-specific limits.

Return type
list of *keystoneclient.v3.limits.Limit*

resource_class

alias of *Limit*

update(*limit*, *project=None*, *service=None*, *resource_name=None*, *resource_limit=None*, *description=None*, ***kwargs*)

Update a project-specific limit.

Parameters

- **limit** a limit to update
- **project** (str or *keystoneclient.v3.projects.Project*) the project ID of the limit to update
- **resource_limit** the limit of the limits resource to update

- **description** (*str*) a description of the limit

Type
resource_limit: int

Returns
a reference of the updated limit.

Return type
keystoneclient.v3.limits.Limit

keystoneclient.v3.policies module

class keystoneclient.v3.policies.**Policy**(*manager*, *info*, *loaded=False*)

Bases: *Resource*

Represents an Identity policy.

Attributes:

- id: a uuid that identifies the policy
- blob: a policy document (blob)
- type: the MIME type of the policy blob

update(*blob=None*, *type=None*)

class keystoneclient.v3.policies.**PolicyManager**(*client*)

Bases: *CrudManager*

Manager class for manipulating Identity policies.

collection_key = 'policies'

create(*blob*, *type='application/json'*, ***kwargs*)

Create a policy.

Parameters

- **blob** (*str*) the policy document.
- **type** (*str*) the MIME type of the policy blob.
- **kwargs** any other attribute provided will be passed to the server.

Returns
the created policy returned from server.

Return type
keystoneclient.v3.policies.Policy

delete(*policy*)

Delete a policy.

Parameters
policy (str or *keystoneclient.v3.policies.Policy*) the policy to be deleted on the server.

Returns
Response object with 204 status.

Return type
`requests.models.Response`

`get`(*policy*)

Retrieve a policy.

Parameters
policy (str or *keystoneclient.v3.policies.Policy*) the policy to be retrieved from the server.

Returns
the specified policy returned from server.

Return type
keystoneclient.v3.policies.Policy

`key = 'policy'`

`list`(***kwargs*)

List policies.

Parameters
kwargs allows filter criteria to be passed where supported by the server.

Returns
a list of policies.

Return type
list of *keystoneclient.v3.policies.Policy*.

`resource_class`

alias of *Policy*

`update`(*policy*, *blob=None*, *type=None*, ***kwargs*)

Update a policy.

Parameters

- **policy** (str or *keystoneclient.v3.policies.Policy*) the policy to be updated on the server.
- **blob** (*str*) the new policy document.
- **type** (*str*) the new MIME type of the policy blob.
- **kwargs** any other attribute provided will be passed to the server.

Returns
the updated policy returned from server.

Return type
keystoneclient.v3.policies.Policy

keystoneclient.v3.projects module

class keystoneclient.v3.projects.**Project**(*manager*, *info*, *loaded=False*)

Bases: *Resource*

Represents an Identity project.

Attributes:

- id: a uuid that identifies the project
- name: project name
- description: project description
- enabled: boolean to indicate if project is enabled
- parent_id: a uuid representing this projects parent in hierarchy
- **parents: a list or a structured dict containing the parents of this** project in the hierarchy
- **subtree: a list or a structured dict containing the subtree of this** project in the hierarchy

add_tag(*tag*)

check_tag(*tag*)

delete_all_tags()

delete_tag(*tag*)

list_tags()

update(*name=None*, *description=None*, *enabled=None*)

update_tags(*tags*)

class keystoneclient.v3.projects.**ProjectManager**(*client*)

Bases: *CrudManager*

Manager class for manipulating Identity projects.

add_tag(*project*, *tag*)

Add a tag to a project.

Parameters

- **project** project to add a tag to.
- **tag** str name of tag.

check_tag(*project*, *tag*)

Check if tag is associated with project.

Parameters

- **project** project to check tags for.
- **tag** str name of tag

Returns
true if tag is associated, false otherwise

`collection_key = 'projects'`

`create`(*name*, *domain*, *description=None*, *enabled=True*, *parent=None*, ***kwargs*)

Create a project.

Parameters

- **name** (*str*) the name of the project.
- **domain** (str or *keystoneclient.v3.domains.Domain*) the domain of the project.
- **description** (*str*) the description of the project.
- **enabled** (*bool*) whether the project is enabled.
- **parent** (str or *keystoneclient.v3.projects.Project*) the parent of the project in the hierarchy.
- **kwargs** any other attribute provided will be passed to the server.

Returns
the created project returned from server.

Return type
keystoneclient.v3.projects.Project

`delete`(*project*)

Delete a project.

Parameters
project (str or *keystoneclient.v3.projects.Project*) the project to be deleted on the server.

Returns
Response object with 204 status.

Return type
`requests.models.Response`

`delete_tag`(*project*, *tag*)

Remove tag from project.

Parameters

- **projectd** project to remove tag from.
- **tag** str name of tag to remove from project

`find`(***kwargs*)

Find a single item with attributes matching `**kwargs`.

`get`(*project*, *subtree_as_list=False*, *parents_as_list=False*, *subtree_as_ids=False*, *parents_as_ids=False*)

Retrieve a project.

Parameters

- **project** (str or *keystoneclient.v3.projects.Project*) the project to be retrieved from the server.
- **subtree_as_list** (*bool*) retrieve projects below this project in the hierarchy as a flat list. It only includes the projects in which the current user has role assignments on.
- **parents_as_list** (*bool*) retrieve projects above this project in the hierarchy as a flat list. It only includes the projects in which the current user has role assignments on.
- **subtree_as_ids** (*bool*) retrieve the IDs from the projects below this project in the hierarchy as a structured dictionary.
- **parents_as_ids** (*bool*) retrieve the IDs from the projects above this project in the hierarchy as a structured dictionary.

Returns
the specified project returned from server.

Return type
keystoneclient.v3.projects.Project

Raises
keystoneclient.exceptions.ValidationError if subtree_as_list and subtree_as_ids or parents_as_list and parents_as_ids are included at the same time in the call.

key = 'project'

list(*domain=None*, *user=None*, *parent=None*, ***kwargs*)

List projects.

Parameters

- **domain** (str or *keystoneclient.v3.domains.Domain*) the domain of the projects to be filtered on.
- **user** (str or *keystoneclient.v3.users.User*) filter in projects the specified user has role assignments on.
- **parent** (str or *keystoneclient.v3.projects.Project*) filter in projects the specified project is a parent for
- **kwargs** any other attribute provided will filter projects on. Project tags filter keyword: `tags`, `tags_any`, `not_tags`, and `not_tags_any`. tag attribute type string. Pass in a comma separated string to filter with multiple tags.

Returns
a list of projects.

Return type
list of *keystoneclient.v3.projects.Project*

list_tags(*project*)

List tags associated with project.

Parameters
project project to list tags for.

Returns
list of str tag names

`resource_class`
alias of *Project*

`update`(*project*, *name=None*, *domain=None*, *description=None*, *enabled=None*, ***kwargs*)
Update a project.

Parameters

- **project** (str or *keystoneclient.v3.projects.Project*) the project to be updated on the server.
- **name** (*str*) the new name of the project.
- **domain** (str or *keystoneclient.v3.domains.Domain*) the new domain of the project.
- **description** (*str*) the new description of the project.
- **enabled** (*bool*) whether the project is enabled.
- **kwargs** any other attribute provided will be passed to server.

Returns
the updated project returned from server.

Return type
keystoneclient.v3.projects.Project

`update_tags`(*project*, *tags*)
Update tag list of a project.

Replaces current tag list with list specified in tags parameter.

Parameters

- **project** project to update.
- **tags** list of str tag names to add to the project

Returns
list of tags

keystoneclient.v3.regions module

`class keystoneclient.v3.regions.Region`(*manager*, *info*, *loaded=False*)
Bases: *Resource*

Represents a Catalog region.

Attributes:

- id: a string that identifies the region.
- description: a string that describes the region.
- **parent_region_id: a pre-existing region in the backend or its ID**
  field. Allows for hierarchical region organization.

- enabled: determines whether the endpoint appears in the catalog.

class keystoneclient.v3.regions.**RegionManager**(*client*)

Bases: *CrudManager*

Manager class for manipulating Identity regions.

collection_key = 'regions'

create(*id=None*, *description=None*, *enabled=True*, *parent_region=None*, ***kwargs*)

Create a region.

Parameters

- **id** (*str*) the unique identifier of the region. If not specified an ID will be created by the server.
- **description** (*str*) the description of the region.
- **enabled** (*bool*) whether the region is enabled or not, determining if it appears in the catalog.
- **parent_region** (str or *keystoneclient.v3.regions.Region*) the parent of the region in the hierarchy.
- **kwargs** any other attribute provided will be passed to the server.

Returns
the created region returned from server.

Return type
keystoneclient.v3.regions.Region

delete(*region*)

Delete a region.

Parameters
region (str or *keystoneclient.v3.regions.Region*) the region to be deleted on the server.

Returns
Response object with 204 status.

Return type
requests.models.Response

get(*region*)

Retrieve a region.

Parameters
region (str or *keystoneclient.v3.regions.Region*) the region to be retrieved from the server.

Returns
the specified region returned from server.

Return type
keystoneclient.v3.regions.Region

key = 'region'

list(***kwargs*)

List regions.

Parameters
kwargs any attributes provided will filter regions on.

Returns
a list of regions.

Return type
list of *keystoneclient.v3.regions.Region*.

resource_class

alias of *Region*

update(*region*, *description=None*, *enabled=None*, *parent_region=None*, ***kwargs*)

Update a region.

Parameters

- **region** (str or *keystoneclient.v3.regions.Region*) the region to be updated on the server.
- **description** (*str*) the new description of the region.
- **enabled** (*bool*) determining if the region appears in the catalog by enabling or disabling it.
- **parent_region** (str or *keystoneclient.v3.regions.Region*) the new parent of the region in the hierarchy.
- **kwargs** any other attribute provided will be passed to server.

Returns
the updated region returned from server.

Return type
keystoneclient.v3.regions.Region

keystoneclient.v3.registered_limits module

class keystoneclient.v3.registered_limits.**RegisteredLimit**(*manager*, *info*, *loaded=False*)

Bases: *Resource*

Represents a registered limit.

Attributes:

- id: a UUID that identifies the registered limit
- service_id: a UUID that identifies the service for the limit
- region_id: a UUID that identifies the region for the limit
- resource_name: the name of the resource to limit
- default_limit: the default limit for projects to assume
- description: a description of the registered limit

class keystoneclient.v3.registered_limits.**RegisteredLimitManager**(*client*)

Bases: *CrudManager*

Manager class for registered limits.

collection_key = 'registered_limits'

create(*service*, *resource_name*, *default_limit*, *description=None*, *region=None*, ***kwargs*)

Create a registered limit.

Parameters

- **service** (*str*) a UUID that identifies the service for the limit.
- **resource_name** (*str*) the name of the resource to limit.
- **default_limit** (*int*) the default limit for projects to assume.
- **description** (*str*) a string that describes the limit
- **region** (*str*) a UUID that identifies the region for the limit.

Returns
a reference of the created registered limit.

Return type
keystoneclient.v3.registered_limits.RegisteredLimit

delete(*registered_limit*)

Delete a registered limit.

Parameters
registered_limit (str or *keystoneclient.v3.registered_limits.RegisteredLimit*) the registered limit to delete.

Returns
Response object with 204 status.

Return type
requests.models.Response

get(*registered_limit*)

Retrieve a registered limit.

Parameters
registered_limit (str or *keystoneclient.v3.registered_limits.RegisteredLimit*) the registered limit to get.

Returns
a specific registered limit.

Return type
keystoneclient.v3.registered_limits.RegisteredLimit

key = 'registered_limit'

list(*service=None*, *resource_name=None*, *region=None*, ***kwargs*)

List registered limits.

Any parameter provided will be passed to the server as a filter.

Parameters

- **service** (a UUID or *keystoneclient.v3.services.Service*) filter registered limits by service
- **resource_name** (*str*) filter registered limits by resource name
- **region** (a UUID or *keystoneclient.v3.regions.Region*) filter registered limits by region

Returns
a list of registered limits.

Return type
list of *keystoneclient.v3.registered_limits.RegisteredLimit*

resource_class

alias of *RegisteredLimit*

update(*registered_limit*, *service=None*, *resource_name=None*, *default_limit=None*, *description=None*, *region=None*, ***kwargs*)

Update a registered limit.

Parameters

- **registered_limit** the UUID or reference of the registered limit to update.
- **registered_limit** str or *keystoneclient.v3.registered_limits.RegisteredLimit*
- **service** (*str*) a UUID that identifies the service for the limit.
- **resource_name** (*str*) the name of the resource to limit.
- **default_limit** (*int*) the default limit for projects to assume.
- **description** (*str*) a string that describes the limit
- **region** (*str*) a UUID that identifies the region for the limit.

Returns
a reference of the updated registered limit.

Return type
keystoneclient.v3.registered_limits.RegisteredLimit

keystoneclient.v3.role_assignments module

class keystoneclient.v3.role_assignments.**RoleAssignment**(*manager*, *info*, *loaded=False*)

Bases: *Resource*

Represents an Identity role assignment.

Attributes:

- role: an object which contains a role uuid
- **user or group: an object which contains either a user or** group uuid

- **scope: an object which has either a project or domain object**
  containing an uuid

class keystoneclient.v3.role_assignments.**RoleAssignmentManager**(*client*)

Bases: *CrudManager*

Manager class for manipulating Identity roles assignments.

collection_key = 'role_assignments'

create(***kwargs*)

delete(***kwargs*)

find(***kwargs*)

Find a single item with attributes matching **kwargs.

get(***kwargs*)

key = 'role_assignment'

list(*user=None*, *group=None*, *project=None*, *domain=None*, *system=False*, *role=None*, *effective=False*, *os_inherit_extension_inherited_to=None*, *include_subtree=False*, *include_names=False*)

List role assignments.

If no arguments are provided, all role assignments in the system will be listed.

If both user and group are provided, a ValidationError will be raised. If both domain and project are provided, it will also raise a ValidationError.

Parameters

- **user** User to be used as query filter. (optional)
- **group** Group to be used as query filter. (optional)
- **project** Project to be used as query filter. (optional)
- **domain** Domain to be used as query filter. (optional)
- **system** Boolean to be used to filter system assignments. (optional)
- **role** Role to be used as query filter. (optional)
- **effective** (*boolean*) return effective role assignments. (optional)
- **os_inherit_extension_inherited_to** (*string*) return inherited role assignments for either projects or domains. (optional)
- **include_subtree** (*boolean*) Include subtree (optional)
- **include_names** (*boolean*) Display names instead of IDs. (optional)

put(***kwargs*)

resource_class

alias of *RoleAssignment*

update(***kwargs*)

keystoneclient.v3.roles module

class keystoneclient.v3.roles.**InferenceRule**(*manager*, *info*, *loaded=False*)

Bases: *Resource*

Represents a rule that states one role implies another.

Attributes:

- prior_role: this role implies the other
- implied_role: this role is implied by the other

class keystoneclient.v3.roles.**InferenceRuleManager**(*client*)

Bases: *CrudManager*

Manager class for manipulating Identity inference rules.

check(*prior_role*, *implied_role*)

Check if an inference rule exists.

Valid HTTP return codes:

- 204: The rule inference exists
- 404: A role cannot be found

Parameters

- **prior_role** the role which implies `implied_role`.
- **implied_role** the role which is implied by `prior_role`.

Returns
response object with 204 status returned from server.

Return type
`requests.models.Response`

collection_key = 'role_inferences'

create(*prior_role*, *implied_role*)

Create an inference rule.

An inference rule is comprised of two roles, a prior role and an implied role. The prior role will imply the implied role.

Valid HTTP return codes:

- 201: Resource is created successfully
- 404: A role cannot be found
- 409: The inference rule already exists

Parameters

- **prior_role** the role which implies `implied_role`.
- **implied_role** the role which is implied by `prior_role`.

Returns
a newly created role inference returned from server.

Return type
keystoneclient.v3.roles.InferenceRule

delete(*prior_role*, *implied_role*)

Delete an inference rule.

When deleting an inference rule, both roles are required. Note that neither role is deleted, only the inference relationship is dissolved.

Valid HTTP return codes:

- 204: Delete request is accepted
- 404: A role cannot be found

Parameters

- **prior_role** the role which implies `implied_role`.
- **implied_role** the role which is implied by `prior_role`.

Returns
Response object with 204 status.

Return type
`requests.models.Response`

find(***kwargs*)

Find a single item with attributes matching `**kwargs`.

get(*prior_role*, *implied_role*)

Retrieve an inference rule.

Valid HTTP return codes:

- 200: Inference rule is returned
- 404: A role cannot be found

Parameters

- **prior_role** the role which implies `implied_role`.
- **implied_role** the role which is implied by `prior_role`.

Returns
the specified role inference returned from server.

Return type
keystoneclient.v3.roles.InferenceRule

key = 'role_inference'

list(*prior_role*)

List all roles that a role may imply.

Valid HTTP return codes:

- 200: List of inference rules are returned
- 404: A role cannot be found

Parameters
prior_role the role which implies implied_role.

Returns
the specified role inference returned from server.

Return type
keystoneclient.v3.roles.InferenceRule

list_inference_roles()

List all rule inferences.

Valid HTTP return codes:

- 200: All inference rules are returned

Parameters
kwargs attributes provided will be passed to the server.

Returns
a list of inference rules.

Return type
list of *keystoneclient.v3.roles.InferenceRule*

put(***kwargs*)

resource_class

alias of *InferenceRule*

update(***kwargs*)

class keystoneclient.v3.roles.**Role**(*manager*, *info*, *loaded=False*)

Bases: *Resource*

Represents an Identity role.

Attributes:

- id: a uuid that identifies the role
- name: user-facing identifier
- domain: optional domain for the role

class keystoneclient.v3.roles.**RoleManager**(*client*)

Bases: *CrudManager*

Manager class for manipulating Identity roles.

check(*role*, *user=None*, *group=None*, *system=None*, *domain=None*, *project=None*, *os_inherit_extension_inherited=False*, ***kwargs*)

Check if a user or group has a role on a domain or project.

Parameters

- **user** (str or *keystoneclient.v3.users.User*) check for role grants for the specified user on a resource. Domain or project must be specified. User and group are mutually exclusive.
- **group** (str or *keystoneclient.v3.groups.Group*) check for role grants for the specified group on a resource. Domain or project must be specified. User and group are mutually exclusive.
- **system** (*str*) check for role grants on the system. Project, domain, and system are mutually exclusive.
- **domain** (str or *keystoneclient.v3.domains.Domain*) check for role grants on the specified domain. Either user or group must be specified. Project, domain, and system are mutually exclusive.
- **project** (str or *keystoneclient.v3.projects.Project*) check for role grants on the specified project. Either user or group must be specified. Project, domain, and system are mutually exclusive.
- **os_inherit_extension_inherited** (*bool*) OS-INHERIT will be used. It provides the ability for projects to inherit role assignments from their domains or from parent projects in the hierarchy.
- **kwargs** any other attribute provided will be passed to server.

Returns
the specified role returned from server if it exists.

Return type
keystoneclient.v3.roles.Role

Returns
Response object with 204 status if specified role doesnt exist.

Return type
requests.models.Response

check_implied(*prior_role*, *implied_role*, ***kwargs*)

collection_key = 'roles'

create(*name*, *domain=None*, ***kwargs*)

Create a role.

Parameters

- **name** (*str*) the name of the role.
- **domain** (str or *keystoneclient.v3.domains.Domain*) the domain of the role. If a value is passed it is a domain-scoped role, otherwise its a global role.
- **kwargs** any other attribute provided will be passed to the server.

Returns
the created role returned from server.

Return type
keystoneclient.v3.roles.Role

create_implied(*prior_role*, *implied_role*, ***kwargs*)

delete(*role*)

Delete a role.

When a role is deleted all the role inferences that have deleted role as prior role will be deleted as well.

Parameters

role (str or *keystoneclient.v3.roles.Role*) the role to be deleted on the server.

Returns

Response object with 204 status.

Return type

`requests.models.Response`

delete_implied(*prior_role*, *implied_role*, ***kwargs*)

deprecation_msg = 'keystoneclient.v3.roles.InferenceRuleManager'

get(*role*)

Retrieve a role.

Parameters

role (str or *keystoneclient.v3.roles.Role*) the role to be retrieved from the server.

Returns

the specified role returned from server.

Return type

keystoneclient.v3.roles.Role

get_implied(*prior_role*, *implied_role*, ***kwargs*)

grant(*role*, *user=None*, *group=None*, *system=None*, *domain=None*, *project=None*, *os_inherit_extension_inherited=False*, ***kwargs*)

Grant a role to a user or group on a domain or project.

Parameters

- **role** (str or *keystoneclient.v3.roles.Role*) the role to be granted on the server.
- **user** (str or *keystoneclient.v3.users.User*) the specified user to have the role granted on a resource. Domain or project must be specified. User and group are mutually exclusive.
- **group** (str or *keystoneclient.v3.groups.Group*) the specified group to have the role granted on a resource. Domain or project must be specified. User and group are mutually exclusive.
- **system** (*str*) system information to grant the role on. Project, domain, and system are mutually exclusive.
- **domain** (str or *keystoneclient.v3.domains.Domain*) the domain in which the role will be granted. Either user or group must be specified. Project, domain, and system are mutually exclusive.

- **project** (str or *keystoneclient.v3.projects.Project*) the project in which the role will be granted. Either user or group must be specified. Project, domain, and system are mutually exclusive.
- **os_inherit_extension_inherited** (*bool*) OS-INHERIT will be used. It provides the ability for projects to inherit role assignments from their domains or from parent projects in the hierarchy.
- **kwargs** any other attribute provided will be passed to server.

Returns
the granted role returned from server.

Return type
keystoneclient.v3.roles.Role

key = 'role'

list(*user=None*, *group=None*, *system=None*, *domain=None*, *project=None*, *os_inherit_extension_inherited=False*, ***kwargs*)

List roles and role grants.

Parameters

- **user** (str or *keystoneclient.v3.users.User*) filter in role grants for the specified user on a resource. Domain or project must be specified. User and group are mutually exclusive.
- **group** (str or *keystoneclient.v3.groups.Group*) filter in role grants for the specified group on a resource. Domain or project must be specified. User and group are mutually exclusive.
- **domain** (str or *keystoneclient.v3.domains.Domain*) filter in role grants on the specified domain. Either user or group must be specified. Project, domain, and system are mutually exclusive.
- **project** (str or *keystoneclient.v3.projects.Project*) filter in role grants on the specified project. Either user or group must be specified. Project, domain and system are mutually exclusive.
- **os_inherit_extension_inherited** (*bool*) OS-INHERIT will be used. It provides the ability for projects to inherit role assignments from their domains or from parent projects in the hierarchy.
- **kwargs** any other attribute provided will filter roles on.

Returns
a list of roles.

Return type
list of *keystoneclient.v3.roles.Role*

list_role_inferences(***kwargs*)

resource_class

alias of *Role*

revoke(*role*, *user=None*, *group=None*, *system=None*, *domain=None*, *project=None*, *os_inherit_extension_inherited=False*, ***kwargs*)

Revoke a role from a user or group on a domain or project.

Parameters

- **user** (str or *keystoneclient.v3.users.User*) revoke role grants for the specified user on a resource. Domain or project must be specified. User and group are mutually exclusive.
- **group** (str or *keystoneclient.v3.groups.Group*) revoke role grants for the specified group on a resource. Domain or project must be specified. User and group are mutually exclusive.
- **system** (*str*) revoke role grants on the system. Project, domain, and system are mutually exclusive.
- **domain** (str or *keystoneclient.v3.domains.Domain*) revoke role grants on the specified domain. Either user or group must be specified. Project, domain, and system are mutually exclusive.
- **project** (str or *keystoneclient.v3.projects.Project*) revoke role grants on the specified project. Either user or group must be specified. Project, domain, and system are mutually exclusive.
- **os_inherit_extension_inherited** (*bool*) OS-INHERIT will be used. It provides the ability for projects to inherit role assignments from their domains or from parent projects in the hierarchy.
- **kwargs** any other attribute provided will be passed to server.

Returns

the revoked role returned from server.

Return type

list of *keystoneclient.v3.roles.Role*

update(*role*, *name=None*, ***kwargs*)

Update a role.

Parameters

- **role** (str or *keystoneclient.v3.roles.Role*) the role to be updated on the server.
- **name** (*str*) the new name of the role.
- **kwargs** any other attribute provided will be passed to server.

Returns

the updated role returned from server.

Return type

keystoneclient.v3.roles.Role

keystoneclient.v3.services module

class keystoneclient.v3.services.**Service**(*manager*, *info*, *loaded=False*)

Bases: *Resource*

Represents an Identity service.

Attributes:

- id: a uuid that identifies the service
- name: the user-facing name of the service (e.g. Keystone)
- description: a description of the service
- type: the type of the service (e.g. compute, identity)
- enabled: determines whether the service appears in the catalog

class keystoneclient.v3.services.**ServiceManager**(*client*)

Bases: *CrudManager*

Manager class for manipulating Identity services.

collection_key = 'services'

create(*name*, *type=None*, *enabled=True*, *description=None*, ***kwargs*)

Create a service.

Parameters

- **name** (*str*) the name of the service.
- **type** (*str*) the type of the service.
- **enabled** (*bool*) whether the service appears in the catalog.
- **description** (*str*) the description of the service.
- **kwargs** any other attribute provided will be passed to the server.

Returns
the created service returned from server.

Return type
keystoneclient.v3.services.Service

delete(*service=None*, *id=None*)

Delete a service.

Parameters
service (str or *keystoneclient.v3.services.Service*) the service to be deleted on the server.

Returns
Response object with 204 status.

Return type
requests.models.Response

get(*service*)

Retrieve a service.

Parameters

service (str or *keystoneclient.v3.services.Service*) the service to be retrieved from the server.

Returns

the specified service returned from server.

Return type

keystoneclient.v3.services.Service

key = 'service'

list(*name=None*, *type=None*, ***kwargs*)

List services.

Parameters

- **name** (*str*) the name of the services to be filtered on.
- **type** (*str*) the type of the services to be filtered on.
- **kwargs** any other attribute provided will filter services on.

Returns

a list of services.

Return type

list of *keystoneclient.v3.services.Service*

resource_class

alias of *Service*

update(*service*, *name=None*, *type=None*, *enabled=None*, *description=None*, ***kwargs*)

Update a service.

Parameters

- **service** (str or *keystoneclient.v3.services.Service*) the service to be updated on the server.
- **name** (*str*) the new name of the service.
- **type** (*str*) the new type of the service.
- **enabled** (*bool*) whether the service appears in the catalog.
- **description** (*str*) the new description of the service.
- **kwargs** any other attribute provided will be passed to server.

Returns

the updated service returned from server.

Return type

keystoneclient.v3.services.Service

keystoneclient.v3.system module

class keystoneclient.v3.system.**System**(*manager*, *info*, *loaded=False*)

Bases: *Resource*

Represents the deployment system, with all the services in it.

Attributes:

- all: boolean

keystoneclient.v3.tokens module

class keystoneclient.v3.tokens.**TokenManager**(*client*)

Bases: object

Manager class for manipulating Identity tokens.

get_revoked(*audit_id_only=False*)

Get revoked tokens list.

Parameters

audit_id_only (*bool*) If true, the server is requested to not send token IDs, but only audit IDs instead. **New in version 2.2.0.**

Returns

A dict containing `signed` which is a CMS formatted string if the server signed the response. If *audit_id_only* is true then the response may be a dict containing `revoked` which is the list of token audit IDs and expiration times.

Return type

dict

get_token_data(*token*, *include_catalog=True*, *allow_expired=False*, *access_rules_support=None*)

Fetch the data about a token from the identity server.

Parameters

- **token** (*str*) The ID of the token to be fetched.
- **include_catalog** (*bool*) Whether the service catalog should be included in the response.
- **allow_expired** If True the token will be validated and returned if it has already expired.
- **access_rules_support** (*float*) Version number indicating that the client is capable of enforcing keystone access rules, if unset this client does not support access rules.

Return type

dict

revoke_token(*token*)

Revoke a token.

Parameters
token (str or *keystoneclient.access.AccessInfo*) The token to be revoked.

validate(*token*, *include_catalog=True*, *allow_expired=False*, *access_rules_support=None*)

Validate a token.

Parameters

- **token** (str or *keystoneclient.access.AccessInfo*) The token to be validated.
- **include_catalog** If False, the response is requested to not include the catalog.
- **allow_expired** (*bool*) If True the token will be validated and returned if it has already expired.
- **access_rules_support** (*float*) Version number indicating that the client is capable of enforcing keystone access rules, if unset this client does not support access rules.

Return type
keystoneclient.access.AccessInfoV3

keystoneclient.v3.users module

class keystoneclient.v3.users.**User**(*manager*, *info*, *loaded=False*)

Bases: *Resource*

Represents an Identity user.

Attributes:

- id: a uuid that identifies the user

class keystoneclient.v3.users.**UserManager**(*client*)

Bases: *CrudManager*

Manager class for manipulating Identity users.

add_to_group(*user*, *group*)

Add the specified user as a member of the specified group.

Parameters

- **user** (str or *keystoneclient.v3.users.User*) the user to be added to the group.
- **group** (str or *keystoneclient.v3.groups.Group*) the group to put the user in.

Returns
Response object with 204 status.

Return type
requests.models.Response

check_in_group(*user*, *group*)

Check if the specified user is a member of the specified group.

Parameters

- **user** (str or *keystoneclient.v3.users.User*) the user to be verified in the group.
- **group** (str or *keystoneclient.v3.groups.Group*) the group to check the user in.

Returns
Response object with 204 status.

Return type
`requests.models.Response`

collection_key = 'users'

create(*name*, *domain=None*, *project=None*, *password=None*, *email=None*, *description=None*, *enabled=True*, *default_project=None*, ***kwargs*)

Create a user.

Parameters

- **name** (*str*) the name of the user.
- **domain** (str or *keystoneclient.v3.domains.Domain*) the domain of the user.
- **project** (str or *keystoneclient.v3.projects.Project*) the default project of the user. (deprecated, see warning below)
- **password** (*str*) the password for the user.
- **email** (*str*) the email address of the user.
- **description** (*str*) a description of the user.
- **enabled** (*bool*) whether the user is enabled.
- **default_project** (str or *keystoneclient.v3.projects.Project*) the default project of the user.
- **kwargs** any other attribute provided will be passed to the server.

Returns
the created user returned from server.

Return type
keystoneclient.v3.users.User

> **Warning:** The project argument is deprecated as of the 1.7.0 release in favor of default_project and may be removed in the 2.0.0 release.
>
> If both default_project and project is provided, the default_project will be used.

delete(*user*)

Delete a user.

Parameters
user (str or *keystoneclient.v3.users.User*) the user to be deleted on the server.

Returns
Response object with 204 status.

Return type
requests.models.Response

get(*user*)

Retrieve a user.

Parameters
user (str or *keystoneclient.v3.users.User*) the user to be retrieved from the server.

Returns
the specified user returned from server.

Return type
keystoneclient.v3.users.User

key = 'user'

list(*project=None*, *domain=None*, *group=None*, *default_project=None*, ***kwargs*)

List users.

Parameters

- **project** (str or *keystoneclient.v3.projects.Project*) the default project of the users to be filtered on. (deprecated, see warning below)
- **domain** (str or *keystoneclient.v3.domains.Domain*) the domain of the users to be filtered on.
- **group** (str or *keystoneclient.v3.groups.Group*) the group in which the users are member of.
- **default_project** (str or *keystoneclient.v3.projects.Project*) the default project of the users to be filtered on.
- **kwargs** any other attribute provided will filter users on.

Returns
a list of users.

Return type
list of *keystoneclient.v3.users.User*.

> **Warning:** The project argument is deprecated as of the 1.7.0 release in favor of default_project and may be removed in the 2.0.0 release.
>
> If both default_project and project is provided, the default_project will be used.

remove_from_group(*user*, *group*)

Remove the specified user from the specified group.

Parameters

- **user** (str or *keystoneclient.v3.users.User*) the user to be removed from the group.
- **group** (str or *keystoneclient.v3.groups.Group*) the group to remove the user from.

Returns
Response object with 204 status.

Return type
requests.models.Response

resource_class
alias of *User*

update(*user*, *name=None*, *domain=None*, *project=None*, *password=None*, *email=None*, *description=None*, *enabled=None*, *default_project=None*, ***kwargs*)

Update a user.

Parameters

- **user** (str or *keystoneclient.v3.users.User*) the user to be updated on the server.
- **name** (*str*) the new name of the user.
- **domain** (str or *keystoneclient.v3.domains.Domain*) the new domain of the user.
- **project** (str or *keystoneclient.v3.projects.Project*) the new default project of the user. (deprecated, see warning below)
- **password** (*str*) the new password of the user.
- **email** (*str*) the new email of the user.
- **description** (*str*) the newdescription of the user.
- **enabled** (*bool*) whether the user is enabled.
- **default_project** (str or *keystoneclient.v3.projects.Project*) the new default project of the user.
- **kwargs** any other attribute provided will be passed to server.

Returns
the updated user returned from server.

Return type
keystoneclient.v3.users.User

Warning: The project argument is deprecated as of the 1.7.0 release in favor of default_project and may be removed in the 2.0.0 release.

If both default_project and project is provided, the default_project will be used.

update_password(*old_password*, *new_password*)

Update the password for the user the token belongs to.

Parameters

- **old_password** (*str*) the users old password
- **new_password** (*str*) the users new password

Returns
Response object with 204 status.

Return type
requests.models.Response

Module contents

4.1.2 Submodules

4.1.3 keystoneclient.access module

class keystoneclient.access.**AccessInfo**(**args*, ***kwargs*)

Bases: dict

Encapsulates a raw authentication token from keystone.

Provides helper methods for extracting useful values from that token.

property audit_chain_id

Return the audit chain ID if present.

In the event that a token was rescoped then this ID will be the *audit_id* of the initial token. Returns None if no value present.

Returns
str or None.

property audit_id

Return the audit ID if present.

Returns
str or None.

property auth_token

Return the token_id associated with the auth request.

To be used in headers for authenticating OpenStack API requests.

Returns
str

property auth_url

Return a tuple of identity URLs.

The identity URLs are from publicURL and adminURL for the service identity from the service catalog associated with the authorization request. If the authentication request wasnt scoped to a tenant (project), this property will return None.

DEPRECATED: this doesnt correctly handle region name. You should fetch it from the service catalog yourself. This may be removed in the 2.0.0 release.

Returns
tuple of urls

property domain_id

Return the domain id associated with the auth request.

Returns
str or None (if no domain associated with the token)

property domain_name

Return the domain name associated with the auth request.

Returns
str or None (if no domain associated with the token)

property domain_scoped

Return true if the auth token was scoped to a domain.

Returns
bool

property expires

Return the token expiration (as datetime object).

Returns
datetime

classmethod factory(*resp=None*, *body=None*, *region_name=None*, *auth_token=None*, ***kwargs*)

Factory function to create a new AccessInfo object.

Create AccessInfo object given a successful auth response & body or a user-provided dict.

Warning: Use of the region_name argument is deprecated as of the 1.7.0 release and may be removed in the 2.0.0 release.

has_service_catalog()

Return true if the authorization token has a service catalog.

Returns
boolean

property initial_audit_id

The audit ID of the initially requested token.

This is the *audit_chain_id* if present or the *audit_id*.

property is_federated

Return true if federation was used to get the token.

Returns
boolean

classmethod is_valid(*body*, ***kwargs*)

Determine if processing valid v2 or v3 token.

Validates from the auth body or a user-provided dict.

Returns
true if auth body matches implementing class

Return type
boolean

property issued

Return the token issue time (as datetime object).

Returns
datetime

property management_url

Return the first adminURL of the identity endpoint.

The identity endpoint is from the service catalog associated with the authorization request, or None if the authentication request wasnt scoped to a tenant (project).

DEPRECATED: this doesnt correctly handle region name. You should fetch it from the service catalog yourself. This may be removed in the 2.0.0 release.

Returns
tuple of urls

property oauth_access_token_id

Return the access token ID if OAuth authentication used.

Returns
str or None.

property oauth_consumer_id

Return the consumer ID if OAuth authentication used.

Returns
str or None.

property project_domain_id

Return the projects domain id associated with the auth request.

For v2, it returns default if a project is scoped or None which may be different from the keystone configuration.

Returns
str

property project_domain_name

Return the projects domain name associated with the auth request.

For v2, it returns Default if a project is scoped or None which may be different from the keystone configuration.

Returns
str

property project_id

Return the project ID associated with the auth request.

This returns None if the auth token wasnt scoped to a project.

Returns
str or None (if no project associated with the token)

property project_name

Return the project name associated with the auth request.

Returns
str or None (if no project associated with the token)

property project_scoped

Return true if the auth token was scoped to a tenant(project).

Returns
bool

property role_ids

Return a list of users role ids associated with the auth request.

Returns
a list of strings of role ids

property role_names

Return a list of users role names associated with the auth request.

Returns
a list of strings of role names

property scoped

Return true if the auth token was scoped.

Return true if scoped to a tenant(project) or domain, and contains a populated service catalog.

> **Warning:** This is deprecated as of the 1.7.0 release in favor of project_scoped and may be removed in the 2.0.0 release.

Returns
bool

property tenant_id

Synonym for project_id.

property tenant_name

Synonym for project_name.

property trust_id

Return the trust id associated with the auth request.

Returns
str or None (if no trust associated with the token)

property trust_scoped

Return true if the auth token was scoped from a delegated trust.

The trust delegation is via the OS-TRUST v3 extension.

Returns
bool

property trustee_user_id

Return the trustee user id associated with a trust.

Returns
str or None (if no trust associated with the token)

property trustor_user_id

Return the trustor user id associated with a trust.

Returns
str or None (if no trust associated with the token)

property user_domain_id

Return the users domain id associated with the auth request.

For v2, it always returns default which may be different from the Keystone configuration.

Returns
str

property user_domain_name

Return the users domain name associated with the auth request.

For v2, it always returns Default which may be different from the Keystone configuration.

Returns
str

property user_id

Return the user id associated with the auth request.

Returns
str

property username

Return the username associated with the auth request.

Follows the pattern defined in the V2 API of first looking for name, returning that if available, and falling back to username if name is unavailable.

Returns
str

property version

Return the version of the auth token from identity service.

Returns
str

will_expire_soon(*stale_duration=None*)

Determine if expiration is about to occur.

Returns
true if expiration is within the given duration

Return type
boolean

class keystoneclient.access.**AccessInfoV2**(**args*, ***kwargs*)

Bases: *AccessInfo*

An object for encapsulating raw v2 auth token from identity service.

property audit_chain_id

Return the audit chain ID if present.

In the event that a token was rescoped then this ID will be the *audit_id* of the initial token. Returns None if no value present.

Returns
str or None.

property audit_id

Return the audit ID if present.

Returns
str or None.

property auth_token

Return the token_id associated with the auth request.

To be used in headers for authenticating OpenStack API requests.

Returns
str

property auth_url

Deprecated as of the 1.7.0 release.

Use service_catalog.get_urls() instead. It may be removed in the 2.0.0 release.

property domain_id

Return the domain id associated with the auth request.

Returns
str or None (if no domain associated with the token)

property domain_name

Return the domain name associated with the auth request.

Returns
str or None (if no domain associated with the token)

property domain_scoped

Return true if the auth token was scoped to a domain.

Returns
bool

property expires

Return the token expiration (as datetime object).

Returns
datetime

`has_service_catalog()`

Return true if the authorization token has a service catalog.

Returns
boolean

`property is_federated`

Return true if federation was used to get the token.

Returns
boolean

`classmethod is_valid(body, **kwargs)`

Determine if processing valid v2 or v3 token.

Validates from the auth body or a user-provided dict.

Returns
true if auth body matches implementing class

Return type
boolean

`property issued`

Return the token issue time (as datetime object).

Returns
datetime

`property management_url`

Deprecated as of the 1.7.0 release.

Use service_catalog.get_urls() instead. It may be removed in the 2.0.0 release.

`property oauth_access_token_id`

Return the access token ID if OAuth authentication used.

Returns
str or None.

`property oauth_consumer_id`

Return the consumer ID if OAuth authentication used.

Returns
str or None.

`property project_domain_id`

Return the projects domain id associated with the auth request.

For v2, it returns default if a project is scoped or None which may be different from the keystone configuration.

Returns
str

property project_domain_name

Return the projects domain name associated with the auth request.

For v2, it returns Default if a project is scoped or None which may be different from the keystone configuration.

Returns
str

property project_id

Return the project ID associated with the auth request.

This returns None if the auth token wasnt scoped to a project.

Returns
str or None (if no project associated with the token)

property project_name

Return the project name associated with the auth request.

Returns
str or None (if no project associated with the token)

property project_scoped

Return true if the auth token was scoped to a tenant(project).

Returns
bool

property role_ids

Return a list of users role ids associated with the auth request.

Returns
a list of strings of role ids

property role_names

Return a list of users role names associated with the auth request.

Returns
a list of strings of role names

property scoped

Deprecated as of the 1.7.0 release.

Use project_scoped instead. It may be removed in the 2.0.0 release.

property trust_id

Return the trust id associated with the auth request.

Returns
str or None (if no trust associated with the token)

property trust_scoped

Return true if the auth token was scoped from a delegated trust.

The trust delegation is via the OS-TRUST v3 extension.

Returns
bool

property trustee_user_id

Return the trustee user id associated with a trust.

Returns
str or None (if no trust associated with the token)

property trustor_user_id

Return the trustor user id associated with a trust.

Returns
str or None (if no trust associated with the token)

property user_domain_id

Return the users domain id associated with the auth request.

For v2, it always returns default which may be different from the Keystone configuration.

Returns
str

property user_domain_name

Return the users domain name associated with the auth request.

For v2, it always returns Default which may be different from the Keystone configuration.

Returns
str

property user_id

Return the user id associated with the auth request.

Returns
str

property username

Return the username associated with the auth request.

Follows the pattern defined in the V2 API of first looking for name, returning that if available, and falling back to username if name is unavailable.

Returns
str

class keystoneclient.access.**AccessInfoV3**(*token*, **args*, ***kwargs*)

Bases: *AccessInfo*

An object encapsulating raw v3 auth token from identity service.

property audit_chain_id

Return the audit chain ID if present.

In the event that a token was rescoped then this ID will be the *audit_id* of the initial token. Returns None if no value present.

Returns
str or None.

property audit_id

Return the audit ID if present.

Returns
str or None.

property auth_url

Deprecated as of the 1.7.0 release.

Use service_catalog.get_urls() instead. It may be removed in the 2.0.0 release.

property domain_id

Return the domain id associated with the auth request.

Returns
str or None (if no domain associated with the token)

property domain_name

Return the domain name associated with the auth request.

Returns
str or None (if no domain associated with the token)

property domain_scoped

Return true if the auth token was scoped to a domain.

Returns
bool

property expires

Return the token expiration (as datetime object).

Returns
datetime

has_service_catalog()

Return true if the authorization token has a service catalog.

Returns
boolean

property is_federated

Return true if federation was used to get the token.

Returns
boolean

classmethod is_valid(*body*, ***kwargs*)

Determine if processing valid v2 or v3 token.

Validates from the auth body or a user-provided dict.

Returns
true if auth body matches implementing class

Return type
boolean

property issued

Return the token issue time (as datetime object).

Returns
datetime

property management_url

Deprecated as of the 1.7.0 release.

Use service_catalog.get_urls() instead. It may be removed in the 2.0.0 release.

property oauth_access_token_id

Return the access token ID if OAuth authentication used.

Returns
str or None.

property oauth_consumer_id

Return the consumer ID if OAuth authentication used.

Returns
str or None.

property project_domain_id

Return the projects domain id associated with the auth request.

For v2, it returns default if a project is scoped or None which may be different from the keystone configuration.

Returns
str

property project_domain_name

Return the projects domain name associated with the auth request.

For v2, it returns Default if a project is scoped or None which may be different from the keystone configuration.

Returns
str

property project_id

Return the project ID associated with the auth request.

This returns None if the auth token wasnt scoped to a project.

Returns
str or None (if no project associated with the token)

property project_name

Return the project name associated with the auth request.

Returns
str or None (if no project associated with the token)

property project_scoped

Return true if the auth token was scoped to a tenant(project).

Returns
bool

property role_ids

Return a list of users role ids associated with the auth request.

Returns
a list of strings of role ids

property role_names

Return a list of users role names associated with the auth request.

Returns
a list of strings of role names

property scoped

Deprecated as of the 1.7.0 release.

Use project_scoped instead. It may be removed in the 2.0.0 release.

property trust_id

Return the trust id associated with the auth request.

Returns
str or None (if no trust associated with the token)

property trust_scoped

Return true if the auth token was scoped from a delegated trust.

The trust delegation is via the OS-TRUST v3 extension.

Returns
bool

property trustee_user_id

Return the trustee user id associated with a trust.

Returns
str or None (if no trust associated with the token)

property trustor_user_id

Return the trustor user id associated with a trust.

Returns
str or None (if no trust associated with the token)

property user_domain_id

Return the users domain id associated with the auth request.

For v2, it always returns default which may be different from the Keystone configuration.

Returns
str

property user_domain_name

Return the users domain name associated with the auth request.

For v2, it always returns Default which may be different from the Keystone configuration.

Returns
str

property user_id

Return the user id associated with the auth request.

Returns
str

property username

Return the username associated with the auth request.

Follows the pattern defined in the V2 API of first looking for name, returning that if available, and falling back to username if name is unavailable.

Returns

str

4.1.4 keystoneclient.adapter module

class keystoneclient.adapter.**Adapter**(*session*, *service_type=None*, *service_name=None*, *interface=None*, *region_name=None*, *endpoint_override=None*, *version=None*, *auth=None*, *user_agent=None*, *connect_retries=None*, *logger=None*)

Bases: object

An instance of a session with local variables.

A session is a global object that is shared around amongst many clients. It therefore contains state that is relevant to everyone. There is a lot of state such as the service type and region_name that are only relevant to a particular client that is using the session. An adapter provides a wrapper of client local data around the global session object.

Parameters

- **session** (keystoneclient.session.Session) The session object to wrap.
- **service_type** (*str*) The default service_type for URL discovery.
- **service_name** (*str*) The default service_name for URL discovery.
- **interface** (*str*) The default interface for URL discovery.
- **region_name** (*str*) The default region_name for URL discovery.
- **endpoint_override** (*str*) Always use this endpoint URL for requests for this client.
- **version** (*tuple*) The version that this API targets.
- **auth** (keystoneclient.auth.base.BaseAuthPlugin) An auth plugin to use instead of the session one.
- **user_agent** (*str*) The User-Agent string to set.
- **connect_retries** (*int*) the maximum number of retries that should be attempted for connection errors. Default None - use session default which is dont retry.
- **logger** (*logging.Logger*) A logging object to use for requests that pass through this adapter.

delete(*url*, ***kwargs*)

get(*url*, ***kwargs*)

get_endpoint(*auth=None*, ***kwargs*)

Get an endpoint as provided by the auth plugin.

Parameters

auth (*keystoneclient.auth.base.BaseAuthPlugin*) The auth plugin to use for token. Overrides the plugin on the session. (optional)

Raises

keystoneclient.exceptions.MissingAuthPlugin if a plugin is not available.

Returns

An endpoint if available or None.

Return type

string

get_project_id(*auth=None*)

Return the authenticated project_id as provided by the auth plugin.

Parameters

auth (keystoneclient.auth.base.BaseAuthPlugin) The auth plugin to use for token. Overrides the plugin on the session. (optional)

Raises

- *keystoneclient.exceptions.AuthorizationFailure* if a new token fetch fails.
- *keystoneclient.exceptions.MissingAuthPlugin* if a plugin is not available.

Returns

Current *project_id* or None if not supported by plugin.

Return type

string

get_token(*auth=None*)

Return a token as provided by the auth plugin.

Parameters

auth (*keystoneclient.auth.base.BaseAuthPlugin*) The auth plugin to use for token. Overrides the plugin on the session. (optional)

Raises

keystoneclient.exceptions.AuthorizationFailure if a new token fetch fails.

Returns

A valid token.

Return type

string

get_user_id(*auth=None*)

Return the authenticated user_id as provided by the auth plugin.

Parameters
auth (keystoneclient.auth.base.BaseAuthPlugin) The auth plugin to use for token. Overrides the plugin on the session. (optional)

Raises
- *keystoneclient.exceptions.AuthorizationFailure* if a new token fetch fails.
- *keystoneclient.exceptions.MissingAuthPlugin* if a plugin is not available.

Returns
Current *user_id* or None if not supported by plugin.

Return type
string

head(*url*, ***kwargs*)

invalidate(*auth=None*)
Invalidate an authentication plugin.

patch(*url*, ***kwargs*)

post(*url*, ***kwargs*)

put(*url*, ***kwargs*)

request(*url*, *method*, ***kwargs*)

class keystoneclient.adapter.**LegacyJsonAdapter**(*session*, *service_type=None*, *service_name=None*, *interface=None*, *region_name=None*, *endpoint_override=None*, *version=None*, *auth=None*, *user_agent=None*, *connect_retries=None*, *logger=None*)

Bases: *Adapter*

Make something that looks like an old HTTPClient.

A common case when using an adapter is that we want an interface similar to the HTTPClients of old which returned the body as JSON as well.

You probably dont want this if you are starting from scratch.

request(**args*, ***kwargs*)

4.1.5 keystoneclient.base module

Base utilities to build API operation managers and objects on top of.

class keystoneclient.base.**CrudManager**(*client*)

Bases: *Manager*

Base manager class for manipulating Keystone entities.

Children of this class are expected to define a *collection_key* and *key*.

- *collection_key*: Usually a plural noun by convention (e.g. *entities*); used to refer collections in both URLs (e.g. */v3/entities*) and JSON objects containing a list of member resources (e.g. *{entities: [{}, {}, {}]}*).
- *key*: Usually a singular noun by convention (e.g. *entity*); used to refer to an individual member of the collection.

base_url = None

build_key_only_query(*params_list*)

Build a query that does not include values, just keys.

The Identity API has some calls that define queries without values, this can not be accomplished by using urllib.parse.urlencode(). This method builds a query using only the keys.

build_url(*dict_args_in_out=None*)

Build a resource URL for the given kwargs.

Given an example collection where *collection_key* = *entities* and *key* = *entity*, the following URLs could be generated.

By default, the URL will represent a collection of entities, e.g.:

```
/entities
```

If kwargs contains an *entity_id*, then the URL will represent a specific member, e.g.:

```
/entities/{entity_id}
```

If a *base_url* is provided, the generated URL will be appended to it.

If a tail is provided, it will be appended to the end of the URL.

collection_key = None

create(***kwargs*)

delete(***kwargs*)

find(***kwargs*)

Find a single item with attributes matching `**kwargs`.

get(***kwargs*)

head(***kwargs*)

key = None

list(*fallback_to_auth=False*, ***kwargs*)

put(***kwargs*)

update(***kwargs*)

class keystoneclient.base.**Manager**(*client*)

Bases: object

Basic manager type providing common operations.

Managers interact with a particular type of API (servers, flavors, images, etc.) and provide CRUD operations for them.

Parameters
client instance of BaseClient descendant for HTTP requests

property api

The client.

> **Warning:** This property is deprecated as of the 1.7.0 release in favor of `client()` and may be removed in the 2.0.0 release.

resource_class = None

class keystoneclient.base.**ManagerWithFind**(*client*)

Bases: *Manager*

Manager with additional *find()*/*findall()* methods.

find(***kwargs*)

Find a single item with attributes matching `**kwargs`.

This isnt very efficient: it loads the entire list then filters on the Python side.

findall(***kwargs*)

Find all items with attributes matching `**kwargs`.

This isnt very efficient: it loads the entire list then filters on the Python side.

abstract list()

class keystoneclient.base.**Resource**(*manager*, *info*, *loaded=False*)

Bases: object

Base class for OpenStack resources (tenant, user, etc.).

This is pretty much just a bag for attributes.

HUMAN_ID = False

NAME_ATTR = 'name'

delete()

get()

Support for lazy loading details.

Some clients, such as novaclient have the option to lazy load the details, details which can be loaded with this function.

property human_id

Human-readable ID which can be used for bash completion.

is_loaded()

set_loaded(*val*)

to_dict()

class keystoneclient.base.**Response**(*http_response*, *data*)

Bases: object

keystoneclient.base.**filter_kwargs**(*f*)

keystoneclient.base.**filter_none**(***kwargs*)

Remove any entries from a dictionary where the value is None.

keystoneclient.base.**getid**(*obj*)

Return id if argument is a Resource.

Abstracts the common pattern of allowing both an object or an objects ID (UUID) as a parameter when dealing with relationships.

4.1.6 keystoneclient.baseclient module

class keystoneclient.baseclient.**Client**(*session*)

Bases: object

delete(*url*, ***kwargs*)

get(*url*, ***kwargs*)

head(*url*, ***kwargs*)

patch(*url*, ***kwargs*)

post(*url*, ***kwargs*)

put(*url*, ***kwargs*)

request(*url*, *method*, ***kwargs*)

4.1.7 keystoneclient.client module

keystoneclient.client.**Client**(*version=None*, *unstable=False*, *session=None*, ***kwargs*)

Factory function to create a new identity service client.

The returned client will be either a V3 or V2 client. Check the version using the *version* property or the instances class (with instanceof).

Parameters

- **version** (*tuple*) The required version of the identity API. If specified the client will be selected such that the major version is equivalent and an endpoint provides at least the specified minor version. For example to specify the 3.1 API use `(3, 1)`. (optional)
- **unstable** (*bool*) Accept endpoints not marked as stable. (optional)
- **session** (keystoneclient.session.Session) A session object to be used for communication. If one is not provided it will be constructed from the provided kwargs. (optional)
- **kwargs** Additional arguments are passed through to the client that is being created.

Returns

New keystone client object.

Return type

keystoneclient.v3.client.Client or *keystoneclient.v2_0.client.Client*

Raises

- ***keystoneclient.exceptions.DiscoveryFailure*** if the servers response is invalid.
- ***keystoneclient.exceptions.VersionNotAvailable*** if a suitable client cannot be found.

class keystoneclient.client.**HTTPClient**(*username=None*, *tenant_id=None*, *tenant_name=None*, *password=None*, *auth_url=None*, *region_name=None*, *endpoint=None*, *token=None*, *auth_ref=None*, *use_keyring=False*, *force_new_token=False*, *stale_duration=None*, *user_id=None*, *user_domain_id=None*, *user_domain_name=None*, *domain_id=None*, *domain_name=None*, *project_id=None*, *project_name=None*, *project_domain_id=None*, *project_domain_name=None*, *trust_id=None*, *session=None*, *service_name=None*, *interface='default'*, *endpoint_override=None*, *auth=None*, *user_agent='python-keystoneclient'*, *connect_retries=None*, ***kwargs*)

Bases: *HTTPClient*

Deprecated alias for httpclient.HTTPClient.

This class is deprecated as of the 1.7.0 release in favor of *keystoneclient.httpclient.HTTPClient* and may be removed in the 2.0.0 release.

4.1.8 keystoneclient.discover module

class keystoneclient.discover.**Discover**(*session=None*, *authenticated=None*, ***kwargs*)

Bases: Discover

A means to discover and create clients.

Clients are created depending on the supported API versions on the server.

Querying the server is done on object creation and every subsequent method operates upon the data that was retrieved.

The connection parameters associated with this method are the same format and name as those used by a client (see *keystoneclient.v2_0.client.Client* and *keystoneclient.v3.client.Client*). If not overridden in subsequent methods they will also be what is passed to the constructed client.

In the event that auth_url and endpoint is provided then auth_url will be used in accordance with how the client operates.

Warning: Creating an instance of this class without using the session argument is deprecated as of the 1.7.0 release and may be removed in the 2.0.0 release.

Parameters

- **session** (keystoneclient.session.Session) A session object that will be used for communication. Clients will also be constructed with this session.
- **auth_url** (*string*) Identity service endpoint for authorization. (optional)
- **endpoint** (*string*) A user-supplied endpoint URL for the identity service. (optional)
- **original_ip** (*string*) The original IP of the requesting user which will be sent to identity service in a Forwarded header. (optional) This is ignored if a session is provided. Deprecated as of the 1.7.0 release and may be removed in the 2.0.0 release.
- **debug** (*boolean*) Enables debug logging of all request and responses to the identity service. default False (optional) This is ignored if a session is provided. Deprecated as of the 1.7.0 release and may be removed in the 2.0.0 release.
- **cacert** (*string*) Path to the Privacy Enhanced Mail (PEM) file which contains the trusted authority X.509 certificates needed to established SSL connection with the identity service. (optional) This is ignored if a session is provided. Deprecated as of the 1.7.0 release and may be removed in the 2.0.0 release.
- **key** (*string*) Path to the Privacy Enhanced Mail (PEM) file which contains the unencrypted client private key needed to established two-way SSL connec-

tion with the identity service. (optional) This is ignored if a session is provided. Deprecated as of the 1.7.0 release and may be removed in the 2.0.0 release.

- **cert** (*string*) Path to the Privacy Enhanced Mail (PEM) file which contains the corresponding X.509 client certificate needed to established two-way SSL connection with the identity service. (optional) This is ignored if a session is provided. Deprecated as of the 1.7.0 release and may be removed in the 2.0.0 release.
- **insecure** (*boolean*) Does not perform X.509 certificate validation when establishing SSL connection with identity service. default: False (optional) This is ignored if a session is provided. Deprecated as of the 1.7.0 release and may be removed in the 2.0.0 release.
- **authenticated** (*bool*) Should a token be used to perform the initial discovery operations. default: None (attach a token if an auth plugin is available).

available_versions(***kwargs*)

Return a list of identity APIs available on the server.

The list returned includes the data associated with them.

> **Warning:** This method is deprecated as of the 1.7.0 release in favor of *raw_version_data()* and may be removed in the 2.0.0 release.

Parameters

- **unstable** (*bool*) Accept endpoints not marked stable. (optional) Equates to setting allow_experimental and allow_unknown to True.
- **allow_experimental** (*bool*) Allow experimental version endpoints.
- **allow_deprecated** (*bool*) Allow deprecated version endpoints.
- **allow_unknown** (*bool*) Allow endpoints with an unrecognised status.

Returns

A List of dictionaries as presented by the server. Each dict will contain the version and the URL to use for the version. It is a direct representation of the layout presented by the identity API.

create_client(*version=None*, *unstable=False*, ***kwargs*)

Factory function to create a new identity service client.

Parameters

- **version** (*tuple*) The required version of the identity API. If specified the client will be selected such that the major version is equivalent and an endpoint provides at least the specified minor version. For example to specify the 3.1 API use (3, 1). (optional)
- **unstable** (*bool*) Accept endpoints not marked stable. (optional)
- **kwargs** Additional arguments will override those provided to this objects constructor.

Returns
An instantiated identity client object.

Raises

- *keystoneclient.exceptions.DiscoveryFailure* if the server response is invalid
- *keystoneclient.exceptions.VersionNotAvailable* if a suitable client cannot be found.

raw_version_data(*unstable=False*, ***kwargs*)

Get raw version information from URL.

Raw data indicates that only minimal validation processing is performed on the data, so what is returned here will be the data in the same format it was received from the endpoint.

Parameters

- **unstable** (*bool*) equates to setting allow_experimental and allow_unknown. This argument is deprecated as of the 1.7.0 release and may be removed in the 2.0.0 release.
- **allow_experimental** (*bool*) Allow experimental version endpoints.
- **allow_deprecated** (*bool*) Allow deprecated version endpoints.
- **allow_unknown** (*bool*) Allow endpoints with an unrecognised status.

Returns
The endpoints returned from the server that match the criteria.

Return type
List

Example:

```
>>> from keystoneclient import discover
>>> disc = discover.Discovery(auth_url='http://localhost:5000')
>>> disc.raw_version_data()
    [{'id': 'v3.0',
        'links': [{'href': 'http://127.0.0.1:5000/v3/',
                   'rel': 'self'}],
      'media-types': [
          {'base': 'application/json',
           'type': 'application/vnd.openstack.identity-v3+json'},
          {'base': 'application/xml',
           'type': 'application/vnd.openstack.identity-v3+xml'}],
      'status': 'stable',
      'updated': '2013-03-06T00:00:00Z'},
     {'id': 'v2.0',
      'links': [{'href': 'http://127.0.0.1:5000/v2.0/',
                 'rel': 'self'},
                {'href': '...',
                 'rel': 'describedby',
                 'type': 'application/pdf'}],
      'media-types': [
```

(continues on next page)

(continued from previous page)

```
        {'base': 'application/json',
         'type': 'application/vnd.openstack.identity-v2.0+json'},
        {'base': 'application/xml',
         'type': 'application/vnd.openstack.identity-v2.0+xml'}],
    'status': 'stable',
    'updated': '2013-03-06T00:00:00Z'}]
```

keystoneclient.discover.**add_catalog_discover_hack**(*service_type*, *old*, *new*)

Add a version removal rule for a particular service.

Originally deployments of OpenStack would contain a versioned endpoint in the catalog for different services. E.g. an identity service might look like `http://localhost:5000/v2.0`. This is a problem when we want to use a different version like v3.0 as there is no way to tell where it is located. We cannot simply change all service catalogs either so there must be a way to handle the older style of catalog.

This function adds a rule for a given service type that if part of the URL matches a given regular expression in *old* then it will be replaced with the *new* value. This will replace all instances of old with new. It should therefore contain a regex anchor.

For example the included rule states:

```
add_catalog_version_hack('identity', re.compile('/v2.0/?$'), '/')
```

so if the catalog retrieves an *identity* URL that ends with /v2.0 or /v2.0/ then it should replace it simply with / to fix the users catalog.

Parameters

- **service_type** (*str*) The service type as defined in the catalog that the rule will apply to.
- **old** (*re.RegexObject*) The regular expression to search for and replace if found.
- **new** (*str*) The new string to replace the pattern with.

keystoneclient.discover.**available_versions**(*url*, *session=None*, ***kwargs*)

Retrieve raw version data from a url.

keystoneclient.discover.**normalize_version_number**(*version*)

Turn a version representation into a tuple.

Takes a string, tuple or float which represent version formats we can handle and converts them into a (major, minor) version tuple that we can actually use for discovery.

e.g. v3.3 gives (3, 3)

3.1 gives (3, 1)

Parameters

version Inputted version number to try and convert.

Returns

A usable version tuple

Return type
tuple

Raises
`TypeError` if the inputted version cannot be converted to tuple.

`keystoneclient.discover.version_match(required, candidate)`

Test that an available version satisfies the required version.

To be suitable a version must be of the same major version as required and be at least a match in minor/patch level.

eg. 3.3 is a match for a required 3.1 but 4.1 is not.

Parameters

- **required** (*tuple*) the version that must be met.
- **candidate** (*tuple*) the version to test against required.

Returns
True if candidate is suitable False otherwise.

Return type
bool

4.1.9 keystoneclient.exceptions module

Exception definitions.

`exception keystoneclient.exceptions.AmbiguousEndpoints(endpoints=None)`

Bases: `CatalogException`

Found more than one matching endpoint in Service Catalog.

`exception keystoneclient.exceptions.AuthPluginOptionsMissing(opt_names)`

Bases: *AuthorizationFailure*

Auth plugin misses some options.

`exception keystoneclient.exceptions.AuthSystemNotFound(auth_system)`

Bases: *AuthorizationFailure*

User has specified an AuthSystem that is not installed.

`exception keystoneclient.exceptions.AuthorizationFailure(message=None)`

Bases: *ClientException*

message = 'Cannot authorize API client.'

`exception keystoneclient.exceptions.BadGateway(message=None, details=None, response=None, request_id=None, url=None, method=None, http_status=None, retry_after=0)`

Bases: *HttpServerError*

HTTP 502 - Bad Gateway.

The server was acting as a gateway or proxy and received an invalid response from the upstream server.

http_status = 502

message = 'Bad Gateway'

exception keystoneclient.exceptions.**BadRequest**(*message=None*, *details=None*, *response=None*, *request_id=None*, *url=None*, *method=None*, *http_status=None*, *retry_after=0*)

Bases: *HTTPClientError*

HTTP 400 - Bad Request.

The request cannot be fulfilled due to bad syntax.

http_status = 400

message = 'Bad Request'

exception keystoneclient.exceptions.**CMSError**(*output*)

Bases: Exception

Error reading the certificate.

exception keystoneclient.exceptions.**CertificateConfigError**(*output*)

Bases: Exception

Error reading the certificate.

exception keystoneclient.exceptions.**ClientException**(*message=None*)

Bases: Exception

The base exception for everything to do with clients.

message = 'ClientException'

exception keystoneclient.exceptions.**CommandError**(*message=None*)

Bases: *ClientException*

Error in CLI tool.

exception keystoneclient.exceptions.**Conflict**(*message=None*, *details=None*, *response=None*, *request_id=None*, *url=None*, *method=None*, *http_status=None*, *retry_after=0*)

Bases: *HTTPClientError*

HTTP 409 - Conflict.

Indicates that the request could not be processed because of conflict in the request, such as an edit conflict.

http_status = 409

message = 'Conflict'

exception keystoneclient.exceptions.**ConnectionError**(*message=None*)

Bases: *ClientException*

message = 'Cannot connect to API service.'

keystoneclient.exceptions.**ConnectionRefused**

Connection refused while trying to connect to API service.

An alias of keystoneauth1.exceptions.connection.ConnectFailure

exception keystoneclient.exceptions.**DiscoveryFailure**(*message=None*)

Bases: *ClientException*

message = 'Discovery of client versions failed.'

exception keystoneclient.exceptions.**EmptyCatalog**(*message=None*)

Bases: *EndpointNotFound*

message = 'The service catalog is empty.'

keystoneclient.exceptions.**EndpointException**

Something is rotten in Service Catalog.

An alias of keystoneauth1.exceptions.catalog.CatalogException

exception keystoneclient.exceptions.**EndpointNotFound**(*message=None*)

Bases: CatalogException

message = 'Could not find requested endpoint in Service Catalog.'

exception keystoneclient.exceptions.**ExpectationFailed**(*message=None*, *details=None*, *response=None*, *request_id=None*, *url=None*, *method=None*, *http_status=None*, *retry_after=0*)

Bases: *HTTPClientError*

HTTP 417 - Expectation Failed.

The server cannot meet the requirements of the Expect request-header field.

http_status = 417

message = 'Expectation Failed'

exception keystoneclient.exceptions.**Forbidden**(*message=None*, *details=None*, *response=None*, *request_id=None*, *url=None*, *method=None*, *http_status=None*, *retry_after=0*)

Bases: *HTTPClientError*

HTTP 403 - Forbidden.

The request was a valid request, but the server is refusing to respond to it.

http_status = 403

message = 'Forbidden'

exception keystoneclient.exceptions.**GatewayTimeout**(*message=None*, *details=None*, *response=None*, *request_id=None*, *url=None*, *method=None*, *http_status=None*, *retry_after=0*)

Bases: *HttpServerError*

HTTP 504 - Gateway Timeout.

The server was acting as a gateway or proxy and did not receive a timely response from the upstream server.

http_status = 504

message = 'Gateway Timeout'

exception keystoneclient.exceptions.**Gone**(*message=None*, *details=None*, *response=None*, *request_id=None*, *url=None*, *method=None*, *http_status=None*, *retry_after=0*)

Bases: *HTTPClientError*

HTTP 410 - Gone.

Indicates that the resource requested is no longer available and will not be available again.

http_status = 410

message = 'Gone'

exception keystoneclient.exceptions.**HTTPClientError**(*message=None*, *details=None*, *response=None*, *request_id=None*, *url=None*, *method=None*, *http_status=None*, *retry_after=0*)

Bases: *HttpError*

Client-side HTTP error.

Exception for cases in which the client seems to have erred.

message = 'HTTP Client Error'

exception keystoneclient.exceptions.**HTTPRedirection**(*message=None*, *details=None*, *response=None*, *request_id=None*, *url=None*, *method=None*, *http_status=None*, *retry_after=0*)

Bases: *HttpError*

HTTP Redirection.

message = 'HTTP Redirection'

exception keystoneclient.exceptions.**HttpError**(*message=None*, *details=None*, *response=None*, *request_id=None*, *url=None*, *method=None*, *http_status=None*, *retry_after=0*)

Bases: *ClientException*

The base exception class for all HTTP exceptions.

http_status = 0

message = 'HTTP Error'

exception keystoneclient.exceptions.**HttpNotImplemented**(*message=None*, *details=None*, *response=None*, *request_id=None*, *url=None*, *method=None*, *http_status=None*, *retry_after=0*)

Bases: *HttpServerError*

HTTP 501 - Not Implemented.

The server either does not recognize the request method, or it lacks the ability to fulfill the request.

http_status = 501

message = 'Not Implemented'

exception keystoneclient.exceptions.**HttpServerError**(*message=None*, *details=None*, *response=None*, *request_id=None*, *url=None*, *method=None*, *http_status=None*, *retry_after=0*)

Bases: *HttpError*

Server-side HTTP error.

Exception for cases in which the server is aware that it has erred or is incapable of performing the request.

message = 'HTTP Server Error'

exception keystoneclient.exceptions.**HttpVersionNotSupported**(*message=None*, *details=None*, *response=None*, *request_id=None*, *url=None*, *method=None*, *http_status=None*, *retry_after=0*)

Bases: *HttpServerError*

HTTP 505 - HttpVersion Not Supported.

The server does not support the HTTP protocol version used in the request.

http_status = 505

message = 'HTTP Version Not Supported'

exception keystoneclient.exceptions.**InternalServerError**(*message=None*, *details=None*, *response=None*, *request_id=None*, *url=None*, *method=None*, *http_status=None*, *retry_after=0*)

Bases: *HttpServerError*

HTTP 500 - Internal Server Error.

A generic error message, given when no more specific message is suitable.

http_status = 500

message = 'Internal Server Error'

exception keystoneclient.exceptions.**InvalidResponse**(*response*)

Bases: *ClientException*

The response from the server is not valid for this request.

exception keystoneclient.exceptions.**LengthRequired**(*message=None*, *details=None*, *response=None*, *request_id=None*, *url=None*, *method=None*, *http_status=None*, *retry_after=0*)

Bases: *HTTPClientError*

HTTP 411 - Length Required.

The request did not specify the length of its content, which is required by the requested resource.

http_status = 411

message = 'Length Required'

exception keystoneclient.exceptions.**MethodNotAllowed**(*message=None*, *details=None*, *response=None*, *request_id=None*, *url=None*, *method=None*, *http_status=None*, *retry_after=0*)

Bases: *HTTPClientError*

HTTP 405 - Method Not Allowed.

A request was made of a resource using a request method not supported by that resource.

http_status = 405

message = 'Method Not Allowed'

exception keystoneclient.exceptions.**MethodNotImplemented**(*message=None*)

Bases: *ClientException*

Method not implemented by the keystoneclient API.

exception keystoneclient.exceptions.**MissingAuthPlugin**(*message=None*)

Bases: AuthPluginException

message = 'An authenticated request is required but no plugin available.'

exception keystoneclient.exceptions.**MultipleChoices**(*message=None*, *details=None*, *response=None*, *request_id=None*, *url=None*, *method=None*, *http_status=None*, *retry_after=0*)

Bases: *HTTPRedirection*

HTTP 300 - Multiple Choices.

Indicates multiple options for the resource that the client may follow.

http_status = 300

message = 'Multiple Choices'

exception keystoneclient.exceptions.**NoMatchingPlugin**(*name*)

Bases: AuthPluginException

No auth plugins could be created from the parameters provided.

Parameters
name (*str*) The name of the plugin that was attempted to load.

name

The name of the plugin that was attempted to load.

exception keystoneclient.exceptions.**NoUniqueMatch**(*message=None*)

Bases: *ClientException*

Multiple entities found instead of one.

exception keystoneclient.exceptions.**NotAcceptable**(*message=None*, *details=None*, *response=None*, *request_id=None*, *url=None*, *method=None*, *http_status=None*, *retry_after=0*)

Bases: *HTTPClientError*

HTTP 406 - Not Acceptable.

The requested resource is only capable of generating content not acceptable according to the Accept headers sent in the request.

http_status = 406

message = 'Not Acceptable'

exception keystoneclient.exceptions.**NotFound**(*message=None*, *details=None*, *response=None*, *request_id=None*, *url=None*, *method=None*, *http_status=None*, *retry_after=0*)

Bases: *HTTPClientError*

HTTP 404 - Not Found.

The requested resource could not be found but may be available again in the future.

http_status = 404

message = 'Not Found'

exception keystoneclient.exceptions.**PaymentRequired**(*message=None*, *details=None*, *response=None*, *request_id=None*, *url=None*, *method=None*, *http_status=None*, *retry_after=0*)

Bases: *HTTPClientError*

HTTP 402 - Payment Required.

Reserved for future use.

http_status = 402

message = 'Payment Required'

exception keystoneclient.exceptions.**PreconditionFailed**(*message=None*, *details=None*, *response=None*, *request_id=None*, *url=None*, *method=None*, *http_status=None*, *retry_after=0*)

Bases: *HTTPClientError*

HTTP 412 - Precondition Failed.

The server does not meet one of the preconditions that the requester put on the request.

http_status = 412

message = 'Precondition Failed'

exception keystoneclient.exceptions.**ProxyAuthenticationRequired**(*message=None*, *details=None*, *response=None*, *request_id=None*, *url=None*, *method=None*, *http_status=None*, *retry_after=0*)

Bases: *HTTPClientError*

HTTP 407 - Proxy Authentication Required.

The client must first authenticate itself with the proxy.

http_status = 407

message = 'Proxy Authentication Required'

exception keystoneclient.exceptions.**RequestEntityTooLarge**(**args*, ***kwargs*)

Bases: *HTTPClientError*

HTTP 413 - Request Entity Too Large.

The request is larger than the server is willing or able to process.

http_status = 413

message = 'Request Entity Too Large'

exception keystoneclient.exceptions.**RequestTimeout**(*message=None*, *details=None*, *response=None*, *request_id=None*, *url=None*, *method=None*, *http_status=None*, *retry_after=0*)

Bases: *HTTPClientError*

HTTP 408 - Request Timeout.

The server timed out waiting for the request.

http_status = 408

message = 'Request Timeout'

exception keystoneclient.exceptions.**RequestUriTooLong**(*message=None*, *details=None*, *response=None*, *request_id=None*, *url=None*, *method=None*, *http_status=None*, *retry_after=0*)

Bases: *HTTPClientError*

HTTP 414 - Request-URI Too Long.

The URI provided was too long for the server to process.

http_status = 414

message = 'Request-URI Too Long'

exception keystoneclient.exceptions.**RequestedRangeNotSatisfiable**(*message=None*, *details=None*, *response=None*, *request_id=None*, *url=None*, *method=None*, *http_status=None*, *retry_after=0*)

Bases: *HTTPClientError*

HTTP 416 - Requested Range Not Satisfiable.

The client has asked for a portion of the file, but the server cannot supply that portion.

http_status = 416

message = 'Requested Range Not Satisfiable'

exception keystoneclient.exceptions.**SSLError**(*message=None*)

Bases: *ConnectionError*

message = 'An SSL error occurred.'

exception keystoneclient.exceptions.**ServiceUnavailable**(*message=None*, *details=None*, *response=None*, *request_id=None*, *url=None*, *method=None*, *http_status=None*, *retry_after=0*)

Bases: *HttpServerError*

HTTP 503 - Service Unavailable.

The server is currently unavailable.

http_status = 503

message = 'Service Unavailable'

exception keystoneclient.exceptions.**Unauthorized**(*message=None*, *details=None*, *response=None*, *request_id=None*, *url=None*, *method=None*, *http_status=None*, *retry_after=0*)

Bases: *HTTPClientError*

HTTP 401 - Unauthorized.

Similar to 403 Forbidden, but specifically for use when authentication is required and has failed or has not yet been provided.

http_status = 401

message = 'Unauthorized'

exception keystoneclient.exceptions.**UnprocessableEntity**(*message=None*, *details=None*, *response=None*, *request_id=None*, *url=None*, *method=None*, *http_status=None*, *retry_after=0*)

Bases: *HTTPClientError*

HTTP 422 - Unprocessable Entity.

The request was well-formed but was unable to be followed due to semantic errors.

http_status = 422

message = 'Unprocessable Entity'

exception keystoneclient.exceptions.**UnsupportedMediaType**(*message=None*, *details=None*, *response=None*, *request_id=None*, *url=None*, *method=None*, *http_status=None*, *retry_after=0*)

Bases: *HTTPClientError*

HTTP 415 - Unsupported Media Type.

The request entity has a media type which the server or resource does not support.

http_status = 415

message = 'Unsupported Media Type'

exception keystoneclient.exceptions.**UnsupportedParameters**(*names*)

Bases: *ClientException*

A parameter that was provided or returned is not supported.

Parameters
names (*List(str)*) Names of the unsupported parameters.

names

Names of the unsupported parameters.

exception keystoneclient.exceptions.**UnsupportedVersion**(*message=None*)

Bases: *ClientException*

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

exception keystoneclient.exceptions.**ValidationError**(*message=None*)

Bases: *ClientException*

Error in validation on API client side.

exception keystoneclient.exceptions.**VersionNotAvailable**(*message=None*)

Bases: *DiscoveryFailure*

message = 'Discovery failed. Requested version is not available.'

keystoneclient.exceptions.**from_response**(*response*, *method*, *url*)

Return an instance of *HttpError* or subclass based on response.

An alias of keystoneauth1.exceptions.http.from_response()

4.1.10 keystoneclient.httpclient module

OpenStack Client interface. Handles the REST calls and responses.

class keystoneclient.httpclient.**HTTPClient**(*username=None*, *tenant_id=None*, *tenant_name=None*, *password=None*, *auth_url=None*, *region_name=None*, *endpoint=None*, *token=None*, *auth_ref=None*, *use_keyring=False*, *force_new_token=False*, *stale_duration=None*, *user_id=None*, *user_domain_id=None*, *user_domain_name=None*, *domain_id=None*, *domain_name=None*, *project_id=None*, *project_name=None*, *project_domain_id=None*, *project_domain_name=None*, *trust_id=None*, *session=None*, *service_name=None*, *interface='default'*, *endpoint_override=None*, *auth=None*, *user_agent='python-keystoneclient'*, *connect_retries=None*, ***kwargs*)

Bases: *Client*, *BaseAuthPlugin*

HTTP client.

Warning: Creating an instance of this class without using the session argument is deprecated as of the 1.7.0 release and may be removed in the 2.0.0 release.

Parameters

- **user_id** (*string*) User ID for authentication. (optional)
- **username** (*string*) Username for authentication. (optional)
- **user_domain_id** (*string*) Users domain ID for authentication. (optional)
- **user_domain_name** (*string*) Users domain name for authentication. (optional)
- **password** (*string*) Password for authentication. (optional)
- **domain_id** (*string*) Domain ID for domain scoping. (optional)
- **domain_name** (*string*) Domain name for domain scoping. (optional)
- **project_id** (*string*) Project ID for project scoping. (optional)
- **project_name** (*string*) Project name for project scoping. (optional)
- **project_domain_id** (*string*) Projects domain ID for project scoping. (optional)
- **project_domain_name** (*string*) Projects domain name for project scoping. (optional)
- **auth_url** (*string*) Identity service endpoint for authorization.
- **region_name** (*string*) Name of a region to select when choosing an endpoint from the service catalog.

- **timeout** (*integer*) This argument is deprecated as of the 1.7.0 release in favor of session and may be removed in the 2.0.0 release. (optional)
- **endpoint** (*string*) A user-supplied endpoint URL for the identity service. Lazy-authentication is possible for API service calls if endpoint is set at instantiation. (optional)
- **token** (*string*) Token for authentication. (optional)
- **cacert** (*string*) This argument is deprecated as of the 1.7.0 release in favor of session and may be removed in the 2.0.0 release. (optional)
- **key** (*string*) This argument is deprecated as of the 1.7.0 release in favor of session and may be removed in the 2.0.0 release. (optional)
- **cert** (*string*) This argument is deprecated as of the 1.7.0 release in favor of session and may be removed in the 2.0.0 release. (optional)
- **insecure** (*boolean*) This argument is deprecated as of the 1.7.0 release in favor of session and may be removed in the 2.0.0 release. (optional)
- **original_ip** (*string*) This argument is deprecated as of the 1.7.0 release in favor of session and may be removed in the 2.0.0 release. (optional)
- **auth_ref** (*dict*) To allow for consumers of the client to manage their own caching strategy, you may initialize a client with a previously captured auth_reference (token). If there are keyword arguments passed that also exist in auth_ref, the value from the argument will take precedence.
- **use_keyring** (*boolean*) Enables caching auth_ref into keyring. default: False (optional)
- **force_new_token** (*boolean*) Keyring related parameter, forces request for new token. default: False (optional)
- **stale_duration** (*integer*) Gap in seconds to determine if token from keyring is about to expire. default: 30 (optional)
- **tenant_name** (*string*) Tenant name. (optional) The tenant_name keyword argument is deprecated as of the 1.7.0 release in favor of project_name and may be removed in the 2.0.0 release.
- **tenant_id** (*string*) Tenant id. (optional) The tenant_id keyword argument is deprecated as of the 1.7.0 release in favor of project_id and may be removed in the 2.0.0 release.
- **trust_id** (*string*) Trust ID for trust scoping. (optional)
- **session** (keystoneclient.session.Session) A Session object to be used for communicating with the identity service.
- **service_name** (*string*) The default service_name for URL discovery. default: None (optional)
- **interface** (*string*) The default interface for URL discovery. default: admin (v2), public (v3). (optional)
- **endpoint_override** (*string*) Always use this endpoint URL for requests for this client. (optional)

- **auth** (keystoneclient.auth.base.BaseAuthPlugin) An auth plugin to use instead of the session one. (optional)
- **user_agent** (*string*) The User-Agent string to set. default: python-keystoneclient (optional)
- **connect_retries** (*int*) the maximum number of retries that should be attempted for connection errors. Default None - use session default which is dont retry. (optional)

property auth_token

authenticate(*username=None*, *password=None*, *tenant_name=None*, *tenant_id=None*, *auth_url=None*, *token=None*, *user_id=None*, *domain_name=None*, *domain_id=None*, *project_name=None*, *project_id=None*, *user_domain_id=None*, *user_domain_name=None*, *project_domain_id=None*, *project_domain_name=None*, *trust_id=None*, *region_name=None*)

Authenticate user.

Uses the data provided at instantiation to authenticate against the Identity server. This may use either a username and password or token for authentication. If a tenant name or id was provided then the resulting authenticated client will be scoped to that tenant and contain a service catalog of available endpoints.

With the v2.0 API, if a tenant name or ID is not provided, the authentication token returned will be unscoped and limited in capabilities until a fully-scoped token is acquired.

With the v3 API, if a domain name or id was provided then the resulting authenticated client will be scoped to that domain. If a project name or ID is not provided, and the authenticating user has a default project configured, the authentication token returned will be scoped to the default project. Otherwise, the authentication token returned will be unscoped and limited in capabilities until a fully-scoped token is acquired.

With the v3 API, with the OS-TRUST extension enabled, the trust_id can be provided to allow project-specific role delegation between users

If successful, sets the self.auth_ref and self.auth_token with the returned token. If not already set, will also set self.management_url from the details provided in the token.

Returns

`True` if authentication was successful.

Raises

- *keystoneclient.exceptions.AuthorizationFailure* if unable to authenticate or validate the existing authorization token
- **keystoneclient.exceptions.ValueError** if insufficient parameters are used.

If keyring is used, token is retrieved from keyring instead. Authentication will only be necessary if any of the following conditions are met:

- keyring is not used
- if token is not found in keyring
- if token retrieved from keyring is expired or about to expired (as determined by stale_duration)

- if force_new_token is true

delete(*url*, ***kwargs*)

Perform an authenticate DELETE request.

This calls *request()* with `method` set to `DELETE` and an authentication token if one is available.

> **Warning:** *DEPRECATED*: This function is no longer used and is deprecated as of the 1.7.0 release and may be removed in the 2.0.0 release. It was designed to be used by the managers and the managers now receive an adapter so this function is no longer on the standard request path.

deprecated_adapter_variables = {'region_name': None}

deprecated_session_variables = {'cert': None, 'original_ip': None, 'timeout': None, 'verify_cert': 'verify'}

get(*url*, ***kwargs*)

Perform an authenticated GET request.

This calls *request()* with `method` set to `GET` and an authentication token if one is available.

> **Warning:** *DEPRECATED*: This function is no longer used and is deprecated as of the 1.7.0 release and may be removed in the 2.0.0 release. It was designed to be used by the managers and the managers now receive an adapter so this function is no longer on the standard request path.

get_auth_ref_from_keyring(***kwargs*)

Retrieve auth_ref from keyring.

If auth_ref is found in keyring, (keyring_key, auth_ref) is returned. Otherwise, (keyring_key, None) is returned.

Returns
(keyring_key, auth_ref) or (keyring_key, None)

Returns
or (None, None) if use_keyring is not set in the object

get_endpoint(*session*, *interface=None*, ***kwargs*)

Return an endpoint for the client.

There are no required keyword arguments to `get_endpoint` as a plugin implementation should use best effort with the information available to determine the endpoint. However there are certain standard options that will be generated by the clients and should be used by plugins:

- `service_type`: what sort of service is required.
- `service_name`: the name of the service in the catalog.
- `interface`: what visibility the endpoint should have.

- `region_name`: the region the endpoint exists in.

Parameters
session (keystoneclient.session.Session) The session object that the auth_plugin belongs to.

Returns
The base URL that will be used to talk to the required service or None if not available.

Return type
string

get_project_id(*session*, ***kwargs*)

Return the project id that we are authenticated to.

Wherever possible the project id should be inferred from the token however there are certain URLs and other places that require access to the currently authenticated project id.

Parameters
session (keystoneclient.session.Session) A session object so the plugin can make HTTP calls.

Returns
A project identifier or None if one is not available.

Return type
str

get_raw_token_from_identity_service(*auth_url*, *username=None*, *password=None*, *tenant_name=None*, *tenant_id=None*, *token=None*, *user_id=None*, *user_domain_id=None*, *user_domain_name=None*, *domain_id=None*, *domain_name=None*, *project_id=None*, *project_name=None*, *project_domain_id=None*, *project_domain_name=None*, *trust_id=None*)

Authenticate against the Identity API and get a token.

Not implemented here because auth protocols should be API version-specific.

Expected to authenticate or validate an existing authentication reference already associated with the client. Invoking this call *always* makes a call to the Identity service.

Returns
(`resp`, `body`)

get_token(*session*, ***kwargs*)

Obtain a token.

How the token is obtained is up to the plugin. If it is still valid it may be re-used, retrieved from cache or invoke an authentication request against a server.

There are no required kwargs. They are passed directly to the auth plugin and they are implementation specific.

Returning None will indicate that no token was able to be retrieved.

This function is misplaced as it should only be required for auth plugins that use the X-Auth-Token header. However due to the way plugins evolved this method is required and often called to trigger an authentication request on a new plugin.

When implementing a new plugin it is advised that you implement this method, however if you dont require the X-Auth-Token header override the *get_headers* method instead.

Parameters
session (keystoneclient.session.Session) A session object so the plugin can make HTTP calls.

Returns
A token to use.

Return type
string

`get_user_id`(*session*, ***kwargs*)

Return a unique user identifier of the plugin.

Wherever possible the user id should be inferred from the token however there are certain URLs and other places that require access to the currently authenticated user id.

Parameters
session (keystoneclient.session.Session) A session object so the plugin can make HTTP calls.

Returns
A user identifier or None if one is not available.

Return type
str

`has_service_catalog`()

Return True if this client provides a service catalog.

`head`(*url*, ***kwargs*)

Perform an authenticated HEAD request.

This calls *request()* with `method` set to `HEAD` and an authentication token if one is available.

> **Warning:** *DEPRECATED*: This function is no longer used and is deprecated as of the 1.7.0 release and may be removed in the 2.0.0 release. It was designed to be used by the managers and the managers now receive an adapter so this function is no longer on the standard request path.

`property management_url`

`patch`(*url*, ***kwargs*)

Perform an authenticate PATCH request.

This calls *request()* with `method` set to `PATCH` and an authentication token if one is available.

> **Warning:** *DEPRECATED*: This function is no longer used and is deprecated as of the 1.7.0 release and may be removed in the 2.0.0 release. It was designed to be used by the managers and the managers now receive an adapter so this function is no longer on the standard request path.

post(*url*, ***kwargs*)

Perform an authenticate POST request.

This calls *request()* with `method` set to `POST` and an authentication token if one is available.

> **Warning:** *DEPRECATED*: This function is no longer used and is deprecated as of the 1.7.0 release and may be removed in the 2.0.0 release. It was designed to be used by the managers and the managers now receive an adapter so this function is no longer on the standard request path.

process_token(*region_name=None*)

Extract and process information from the new auth_ref.

And set the relevant authentication information.

put(*url*, ***kwargs*)

Perform an authenticate PUT request.

This calls *request()* with `method` set to `PUT` and an authentication token if one is available.

> **Warning:** *DEPRECATED*: This function is no longer used and is deprecated as of the 1.7.0 release and may be removed in the 2.0.0 release. It was designed to be used by the managers and the managers now receive an adapter so this function is no longer on the standard request path.

request(**args*, ***kwargs*)

Send an http request with the specified characteristics.

Wrapper around requests.request to handle tasks such as setting headers, JSON encoding/decoding, and error handling.

> **Warning:** *DEPRECATED*: This function is no longer used. It was designed to be used only by the managers and the managers now receive an adapter so this function is no longer on the standard request path. This may be removed in the 2.0.0 release.

serialize(*entity*)

property service_catalog

Return this clients service catalog.

store_auth_ref_into_keyring(*keyring_key*)

Store auth_ref into keyring.

property tenant_id

Provide read-only backwards compatibility for tenant_id.

> **Warning:** This is deprecated as of the 1.7.0 release in favor of project_id and may be removed in the 2.0.0 release.

property tenant_name

Provide read-only backwards compatibility for tenant_name.

> **Warning:** This is deprecated as of the 1.7.0 release in favor of project_name and may be removed in the 2.0.0 release.

version = None

keystoneclient.httpclient.**USER_AGENT = 'python-keystoneclient'**

Default user agent string.

This property is deprecated as of the 1.7.0 release in favor of keystoneclient.session.USER_AGENT and may be removed in the 2.0.0 release.

keystoneclient.httpclient.**request**(**args*, ***kwargs*)

Make a request.

This function is deprecated as of the 1.7.0 release in favor of *keystoneclient.session.request()* and may be removed in the 2.0.0 release.

4.1.11 keystoneclient.i18n module

oslo.i18n integration module.

See https://docs.openstack.org/oslo.i18n/latest/user/index.html .

4.1.12 keystoneclient.service_catalog module

class keystoneclient.service_catalog.**ServiceCatalog**(*region_name=None*)

Bases: object

Helper methods for dealing with a Keystone Service Catalog.

> **Warning:** Setting region_name is deprecated in favor of passing the region name as a parameter to calls made to the service catalog as of the 1.7.0 release and may be removed in the 2.0.0 release.

classmethod factory(*resource_dict*, *token=None*, *region_name=None*)

Create ServiceCatalog object given an auth token.

> **Warning:** Setting region_name is deprecated in favor of passing the region name as a parameter to calls made to the service catalog as of the 1.7.0 release and may be removed in the 2.0.0 release.

abstract get_data()

Get the raw catalog structure.

Get the version dependent catalog structure as it is presented within the resource.

Returns

list containing raw catalog data entries or None

get_endpoints(*service_type=None*, *endpoint_type=None*, *region_name=None*, *service_name=None*)

Fetch and filter endpoints for the specified service(s).

Returns endpoints for the specified service (or all) containing the specified type (or all) and region (or all) and service name.

If there is no name in the service catalog the service_name check will be skipped. This allows compatibility with services that existed before the name was available in the catalog.

abstract get_token()

Fetch token details from service catalog.

Returns a dictionary containing the following:

```
- `id`: Token's ID
- `expires`: Token's expiration
- `user_id`: Authenticated user's ID
- `tenant_id`: Authorized project's ID
- `domain_id`: Authorized domain's ID
```

abstract get_urls(*attr=None*, *filter_value=None*, *service_type='identity'*, *endpoint_type='publicURL'*, *region_name=None*, *service_name=None*)

Fetch endpoint urls from the service catalog.

Fetch the endpoints from the service catalog for a particular endpoint attribute. If no attribute is given, return the first endpoint of the specified type.

Parameters

- **attr** (*string*) Endpoint attribute name.
- **filter_value** (*string*) Endpoint attribute value.
- **service_type** (*string*) Service type of the endpoint.
- **endpoint_type** (*string*) Type of endpoint. Possible values: public or publicURL, internal or internalURL, admin or adminURL
- **region_name** (*string*) Region of the endpoint.
- **service_name** (*string*) The assigned name of the service.

Returns

tuple of urls or None (if no match found)

property region_name

Region name.

> **Warning:** region_name is deprecated in favor of passing the region name as a parameter to calls made to the service catalog as of the 1.7.0 release and may be removed in the 2.0.0 release.

url_for(*attr=None*, *filter_value=None*, *service_type='identity'*, *endpoint_type='publicURL'*, *region_name=None*, *service_name=None*)

Fetch an endpoint from the service catalog.

Fetch the specified endpoint from the service catalog for a particular endpoint attribute. If no attribute is given, return the first endpoint of the specified type.

Valid endpoint types: *public* or *publicURL*,
internal or *internalURL*, *admin* or adminURL‘

Parameters

- **attr** (*string*) Endpoint attribute name.
- **filter_value** (*string*) Endpoint attribute value.
- **service_type** (*string*) Service type of the endpoint.
- **endpoint_type** (*string*) Type of endpoint.
- **region_name** (*string*) Region of the endpoint.
- **service_name** (*string*) The assigned name of the service.

class keystoneclient.service_catalog.**ServiceCatalogV2**(*resource_dict*, *region_name=None*)

Bases: *ServiceCatalog*

An object for encapsulating the v2 service catalog.

The object is created using raw v2 auth token from Keystone.

get_data()

Get the raw catalog structure.

Get the version dependent catalog structure as it is presented within the resource.

Returns
list containing raw catalog data entries or None

get_token()

Fetch token details from service catalog.

Returns a dictionary containing the following:

```
- `id`: Token's ID
- `expires`: Token's expiration
- `user_id`: Authenticated user's ID
```

(continues on next page)

(continued from previous page)

```
- `tenant_id`: Authorized project's ID
- `domain_id`: Authorized domain's ID
```

get_urls(*attr=None*, *filter_value=None*, *service_type='identity'*, *endpoint_type='publicURL'*, *region_name=None*, *service_name=None*)

Fetch endpoint urls from the service catalog.

Fetch the endpoints from the service catalog for a particular endpoint attribute. If no attribute is given, return the first endpoint of the specified type.

Parameters

- **attr** (*string*) Endpoint attribute name.
- **filter_value** (*string*) Endpoint attribute value.
- **service_type** (*string*) Service type of the endpoint.
- **endpoint_type** (*string*) Type of endpoint. Possible values: public or publicURL, internal or internalURL, admin or adminURL
- **region_name** (*string*) Region of the endpoint.
- **service_name** (*string*) The assigned name of the service.

Returns

tuple of urls or None (if no match found)

classmethod is_valid(*resource_dict*)

class keystoneclient.service_catalog.**ServiceCatalogV3**(*token*, *resource_dict*, *region_name=None*)

Bases: *ServiceCatalog*

An object for encapsulating the v3 service catalog.

The object is created using raw v3 auth token from Keystone.

get_data()

Get the raw catalog structure.

Get the version dependent catalog structure as it is presented within the resource.

Returns

list containing raw catalog data entries or None

get_token()

Fetch token details from service catalog.

Returns a dictionary containing the following:

```
- `id`: Token's ID
- `expires`: Token's expiration
- `user_id`: Authenticated user's ID
- `tenant_id`: Authorized project's ID
- `domain_id`: Authorized domain's ID
```

get_urls(*attr=None*, *filter_value=None*, *service_type='identity'*, *endpoint_type='public'*, *region_name=None*, *service_name=None*)

Fetch endpoint urls from the service catalog.

Fetch the endpoints from the service catalog for a particular endpoint attribute. If no attribute is given, return the first endpoint of the specified type.

Parameters

- **attr** (*string*) Endpoint attribute name.
- **filter_value** (*string*) Endpoint attribute value.
- **service_type** (*string*) Service type of the endpoint.
- **endpoint_type** (*string*) Type of endpoint. Possible values: public or publicURL, internal or internalURL, admin or adminURL
- **region_name** (*string*) Region of the endpoint.
- **service_name** (*string*) The assigned name of the service.

Returns

tuple of urls or None (if no match found)

classmethod is_valid(*resource_dict*)

4.1.13 keystoneclient.session module

class keystoneclient.session.**Session**(*auth=None*, *session=None*, *original_ip=None*, *verify=True*, *cert=None*, *timeout=None*, *user_agent=None*, *redirect=30*)

Bases: object

Maintains client communication state and common functionality.

As much as possible the parameters to this class reflect and are passed directly to the requests library.

Parameters

- **auth** (*keystoneclient.auth.base.BaseAuthPlugin*) An authentication plugin to authenticate the session with. (optional, defaults to None)
- **session** (*requests.Session*) A requests session object that can be used for issuing requests. (optional)
- **original_ip** (*string*) The original IP of the requesting user which will be sent to identity service in a Forwarded header. (optional)
- **verify** The verification arguments to pass to requests. These are of the same form as requests expects, so True or False to verify (or not) against system certificates or a path to a bundle or CA certs to check against or None for requests to attempt to locate and use certificates. (optional, defaults to True)
- **cert** A client certificate to pass to requests. These are of the same form as requests expects. Either a single filename containing both the certificate and key or a tuple containing the path to the certificate then a path to the key. (optional)

- **timeout** (*float*) A timeout to pass to requests. This should be a numerical value indicating some amount (or fraction) of seconds or 0 for no timeout. (optional, defaults to 0)
- **user_agent** (*string*) A User-Agent header string to use for the request. If not provided a default is used. (optional, defaults to python-keystoneclient)
- **redirect** (*int/bool*) Controls the maximum number of redirections that can be followed by a request. Either an integer for a specific count or True/False for forever/never. (optional, default to 30)

DEFAULT_REDIRECT_LIMIT = 30

This property is deprecated as of the 1.7.0 release and may be removed in the 2.0.0 release.

REDIRECT_STATUSES = (301, 302, 303, 305, 307)

This property is deprecated as of the 1.7.0 release and may be removed in the 2.0.0 release.

classmethod construct(*kwargs*)

Handle constructing a session from both old and new arguments.

Support constructing a session from the old *HTTPClient* args as well as the new request-style arguments.

> **Warning:** *DEPRECATED as of 1.7.0*: This function is purely for bridging the gap between older client arguments and the session arguments that they relate to. It is not intended to be used as a generic Session Factory. This function may be removed in the 2.0.0 release.

This function purposefully modifies the input kwargs dictionary so that the remaining kwargs dict can be reused and passed on to other functions without session arguments.

delete(*url*, ***kwargs*)

Perform a DELETE request.

This calls *request()* with `method` set to `DELETE`.

get(*url*, ***kwargs*)

Perform a GET request.

This calls *request()* with `method` set to `GET`.

get_auth_connection_params(*auth=None*, ***kwargs*)

Return auth connection params as provided by the auth plugin.

An auth plugin may specify connection parameters to the request like providing a client certificate for communication.

We restrict the values that may be returned from this function to prevent an auth plugin overriding values unrelated to connection parameters. The values that are currently accepted are:

- *cert*: a path to a client certificate, or tuple of client certificate and key pair that are used with this request.
- *verify*: a boolean value to indicate verifying SSL certificates against the system CAs or a path to a CA file to verify with.

These values are passed to the requests library and further information on accepted values may be found there.

Parameters
auth (keystoneclient.auth.base.BaseAuthPlugin) The auth plugin to use for tokens. Overrides the plugin on the session. (optional)

Raises

- *keystoneclient.exceptions.AuthorizationFailure* if a new token fetch fails.
- *keystoneclient.exceptions.MissingAuthPlugin* if a plugin is not available.
- *keystoneclient.exceptions.UnsupportedParameters* if the plugin returns a parameter that is not supported by this session.

Returns
Authentication headers or None for failure.

Return type
dict

get_auth_headers(*auth=None*, ***kwargs*)

Return auth headers as provided by the auth plugin.

Parameters
auth (*keystoneclient.auth.base.BaseAuthPlugin*) The auth plugin to use for token. Overrides the plugin on the session. (optional)

Raises

- *keystoneclient.exceptions.AuthorizationFailure* if a new token fetch fails.
- *keystoneclient.exceptions.MissingAuthPlugin* if a plugin is not available.

Returns
Authentication headers or None for failure.

Return type
dict

classmethod get_conf_options(*deprecated_opts=None*)

Get oslo_config options that are needed for a *Session*.

These may be useful without being registered for config file generation or to manipulate the options before registering them yourself.

The options that are set are:

cafile
The certificate authority filename.

certfile
The client certificate file to present.

keyfile
The key for the client certificate.

insecure
Whether to ignore SSL verification.

timeout
The max time to wait for HTTP connections.

Parameters
deprecated_opts (*dict*) Deprecated options that should be included in the definition of new options. This should be a dict from the name of the new option to a list of oslo.DeprecatedOpts that correspond to the new option. (optional)

For example, to support the `ca_file` option pointing to the new `cafile` option name:

```
old_opt = oslo_cfg.DeprecatedOpt('ca_file', 'old_group')
deprecated_opts={'cafile': [old_opt]}
```

Returns
A list of oslo_config options.

get_endpoint(*auth=None*, ***kwargs*)

Get an endpoint as provided by the auth plugin.

Parameters
auth (*keystoneclient.auth.base.BaseAuthPlugin*) The auth plugin to use for token. Overrides the plugin on the session. (optional)

Raises
keystoneclient.exceptions.MissingAuthPlugin if a plugin is not available.

Returns
An endpoint if available or None.

Return type
string

get_project_id(*auth=None*)

Return the authenticated project_id as provided by the auth plugin.

Parameters
auth (keystoneclient.auth.base.BaseAuthPlugin) The auth plugin to use for token. Overrides the plugin on the session. (optional)

Raises

- *keystoneclient.exceptions.AuthorizationFailure* if a new token fetch fails.
- *keystoneclient.exceptions.MissingAuthPlugin* if a plugin is not available.

Returns string
Current project_id or None if not supported by plugin.

get_token(*auth=None*)

Return a token as provided by the auth plugin.

Parameters
auth (*keystoneclient.auth.base.BaseAuthPlugin*) The auth plugin to use for token. Overrides the plugin on the session. (optional)

Raises
- ***keystoneclient.exceptions.AuthorizationFailure*** if a new token fetch fails.
- ***keystoneclient.exceptions.MissingAuthPlugin*** if a plugin is not available.

> **Warning:** This method is deprecated as of the 1.7.0 release in favor of *get_auth_headers()* and may be removed in the 2.0.0 release. This method assumes that the only header that is used to authenticate a message is X-Auth-Token which may not be correct.

Returns
A valid token.

Return type
string

get_user_id(*auth=None*)

Return the authenticated user_id as provided by the auth plugin.

Parameters
auth (keystoneclient.auth.base.BaseAuthPlugin) The auth plugin to use for token. Overrides the plugin on the session. (optional)

Raises
- ***keystoneclient.exceptions.AuthorizationFailure*** if a new token fetch fails.
- ***keystoneclient.exceptions.MissingAuthPlugin*** if a plugin is not available.

Returns string
Current user_id or None if not supported by plugin.

head(*url*, ***kwargs*)

Perform a HEAD request.

This calls *request()* with `method` set to `HEAD`.

invalidate(*auth=None*)

Invalidate an authentication plugin.

Parameters
auth (*keystoneclient.auth.base.BaseAuthPlugin*) The auth plugin to invalidate. Overrides the plugin on the session. (optional)

classmethod load_from_cli_options(*args*, ***kwargs*)

Create a *Session* object from CLI arguments.

The CLI arguments must have been registered with *register_cli_options()*.

Parameters
args (*Namespace*) result of parsed arguments.

Returns
A new session object.

Return type
Session

classmethod load_from_conf_options(*conf*, *group*, ***kwargs*)

Create a session object from an oslo_config object.

The options must have been previously registered with register_conf_options.

Parameters

- **conf** (*oslo_config.Cfg*) config object to register with.
- **group** (*string*) The ini group to register options in.
- **kwargs** (*dict*) Additional parameters to pass to session construction.

Returns
A new session object.

Return type
Session

patch(*url*, ***kwargs*)

Perform a PATCH request.

This calls *request()* with `method` set to `PATCH`.

post(*url*, ***kwargs*)

Perform a POST request.

This calls *request()* with `method` set to `POST`.

put(*url*, ***kwargs*)

Perform a PUT request.

This calls *request()* with `method` set to `PUT`.

static register_cli_options(*parser*)

Register the argparse arguments that are needed for a session.

Parameters
parser (*argparse.ArgumentParser*) parser to add to.

classmethod register_conf_options(*conf*, *group*, *deprecated_opts=None*)

Register the oslo_config options that are needed for a session.

The options that are set are:

cafile
The certificate authority filename.

certfile
The client certificate file to present.

keyfile
The key for the client certificate.

insecure
Whether to ignore SSL verification.

timeout
The max time to wait for HTTP connections.

Parameters

- **conf** (*oslo_config.Cfg*) config object to register with.
- **group** (*string*) The ini group to register options in.
- **deprecated_opts** (*dict*) Deprecated options that should be included in the definition of new options. This should be a dict from the name of the new option to a list of oslo.DeprecatedOpts that correspond to the new option. (optional)

  For example, to support the `ca_file` option pointing to the new `cafile` option name:

```
old_opt = oslo_cfg.DeprecatedOpt('ca_file', 'old_group')
deprecated_opts={'cafile': [old_opt]}
```

Returns
The list of options that was registered.

request(*url*, *method*, *json=None*, *original_ip=None*, *user_agent=None*, *redirect=None*, *authenticated=None*, *endpoint_filter=None*, *auth=None*, *requests_auth=None*, *raise_exc=True*, *allow_reauth=True*, *log=True*, *endpoint_override=None*, *connect_retries=0*, *logger=<Logger keystoneclient.session (WARNING)>*, ***kwargs*)

Send an HTTP request with the specified characteristics.

Wrapper around *requests.Session.request* to handle tasks such as setting headers, JSON encoding/decoding, and error handling.

Arguments that are not handled are passed through to the requests library.

Parameters

- **url** (*string*) Path or fully qualified URL of HTTP request. If only a path is provided then endpoint_filter must also be provided such that the base URL can be determined. If a fully qualified URL is provided then endpoint_filter will be ignored.
- **method** (*string*) The http method to use. (e.g. GET, POST)
- **original_ip** (*string*) Mark this request as forwarded for this ip. (optional)
- **headers** (*dict*) Headers to be included in the request. (optional)
- **json** Some data to be represented as JSON. (optional)
- **user_agent** (*string*) A user_agent to use for the request. If present will override one present in headers. (optional)
- **redirect** (*int/bool*) the maximum number of redirections that can be followed by a request. Either an integer for a specific count or True/False for forever/never. (optional)

- **connect_retries** (*int*) the maximum number of retries that should be attempted for connection errors. (optional, defaults to 0 - never retry).
- **authenticated** (*bool*) True if a token should be attached to this request, False if not or None for attach if an auth_plugin is available. (optional, defaults to None)
- **endpoint_filter** (*dict*) Data to be provided to an auth plugin with which it should be able to determine an endpoint to use for this request. If not provided then URL is expected to be a fully qualified URL. (optional)
- **endpoint_override** (*str*) The URL to use instead of looking up the endpoint in the auth plugin. This will be ignored if a fully qualified URL is provided but take priority over an endpoint_filter. (optional)
- **auth** (*keystoneclient.auth.base.BaseAuthPlugin*) The auth plugin to use when authenticating this request. This will override the plugin that is attached to the session (if any). (optional)
- **requests_auth** (requests.auth.AuthBase) A requests library auth plugin that cannot be passed via kwarg because the *auth* kwarg collides with our own auth plugins. (optional)
- **raise_exc** (*bool*) If True then raise an appropriate exception for failed HTTP requests. If False then return the request object. (optional, default True)
- **allow_reauth** (*bool*) Allow fetching a new token and retrying the request on receiving a 401 Unauthorized response. (optional, default True)
- **log** (*bool*) If True then log the request and response data to the debug log. (optional, default True)
- **logger** (*logging.Logger*) The logger object to use to log request and responses. If not provided the keystoneclient.session default logger will be used.
- **kwargs** any other parameter that can be passed to requests.Session.request (such as *headers*). Except: data will be overwritten by the data in json param. allow_redirects is ignored as redirects are handled by the session.

Raises

keystoneclient.exceptions.ClientException For connection failure, or to indicate an error response code.

Returns

The response to the request.

user_agent = None

class keystoneclient.session.**TCPKeepAliveAdapter**(*pool_connections=10*, *pool_maxsize=10*, *max_retries=0*, *pool_block=False*)

Bases: HTTPAdapter

The custom adapter used to set TCP Keep-Alive on all connections.

This Adapter also preserves the default behaviour of Requests which disables Nagles Algorithm. See also: http://blogs.msdn.com/b/windowsazurestorage/archive/2010/06/25/nagle-s-algorithm-is-not-friendly-towards-small-requests.aspx

init_poolmanager(**args*, ***kwargs*)

Initializes a urllib3 PoolManager.

This method should not be called from user code, and is only exposed for use when subclassing the `HTTPAdapter`.

Parameters

- **connections** The number of urllib3 connection pools to cache.
- **maxsize** The maximum number of connections to save in the pool.
- **block** Block when no free connections are available.
- **pool_kwargs** Extra keyword arguments used to initialize the Pool Manager.

keystoneclient.session.**request**(*url*, *method='GET'*, ***kwargs*)

4.1.14 keystoneclient.utils module

keystoneclient.utils.**find_resource**(*manager*, *name_or_id*)

Helper for the _find_* methods.

keystoneclient.utils.**hash_signed_token**(*signed_text*, *mode='md5'*)

keystoneclient.utils.**isotime**(*at=None*, *subsecond=False*)

Stringify time in ISO 8601 format.

keystoneclient.utils.**prompt_for_password**()

Prompt user for password if not provided.

Prompt is used so the password doesnt show up in the bash history.

keystoneclient.utils.**prompt_user_password**()

Prompt user for a password.

Prompt for a password if stdin is a tty.

keystoneclient.utils.**strtime**(*at=None*)

4.1.15 Module contents

CHAPTER

FIVE

RELATED IDENTITY PROJECTS

In addition to creating the Python client library, the Keystone team also provides Identity Service, as well as WSGI Middleware.

CHAPTER

SIX

RELEASE NOTES

Read also the Keystoneclient Release Notes.

CHAPTER

SEVEN

CONTRIBUTING

Code is hosted on OpenDev. Submit bugs to the Keystone project on Launchpad. Submit code to the `openstack/python-keystoneclient` project using Gerrit.

Run tests with `tox`.

CHAPTER

EIGHT

INDICES AND TABLES

- genindex
- modindex
- search