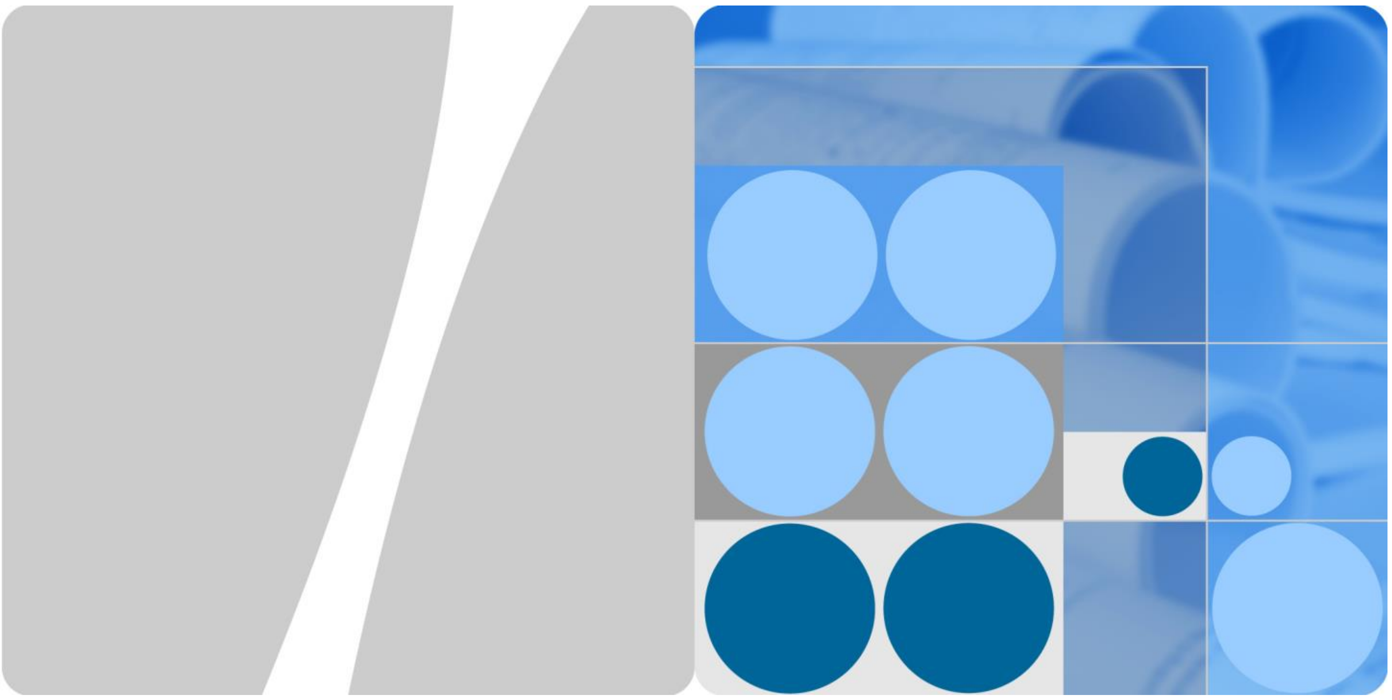


CLI&SDK

Developer Guide

Date **2018-07-30**

Contents

1 Command-Line Interface..5
1.1 CLI Overview..5
1.2 CLI Use Case...9
1.2.1 Creating an ECS Using the Individual CLI 9
1.2.2 Example of Creating an ECS Using the Individual CLI 10
1.2.3 Creating an ECS Using the Unified CLI..12
1.2.4 Use Case for ECS Creation Using Unified CLI..12
1.2.5 Use Case for ECS Deletion Using Individual CLI..13
1.2.6 Use Case for ECS Deletion Using Unified CLI..13
1.3 CLI FAQs...14
1.3.1 Upgrading Software to the Latest Version 14
1.3.2 Rolling Back Software to an Earlier Version 14
1.3.3 Debugging 14
1.3.4 OpenStack Community CLI Reference 14
1.3.5 Cryptography Installation 14
1.3.6 Installing Setuptools and pip Offline 15
1.3.7 How Can I Obtain Online Help?...15
1.3.8 How Can I Obtain the Called APIs? 16
1.3.9 How Can I Obtain CLI Source Code? 16
2 SDK...18
2.1 SDK Overview...18
2.2 Java 18
2.2.1 Java SDK Overview 18
2.2.2 Getting Started...19
2.2.3 Usage 21
2.2.4 IAM Java SDK Demo...23
2.2.5 IMS Java SDK Demo 24
2.2.6 VPC Java SDK Demo...25
2.2.7 ECS Java SDK Demo 27
2.2.8 EVS Java SDK Demo...34
2.2.9 RTS Java SDK Demo 36
2.2.10 AS Java SDK Demo..37

2.2.11 Cloud Eye Java SDK Demo 41
2.2.12 DNS Java SDK Demo 46
2.2.13 ELB Java SDK Demo 49
2.2.14 VBS Java SDK Demo 51
2.2.15 CTS Java SDK Demo 56
2.2.16 SMN Java SDK Demo 57
2.2.17 MaaS Java SDK Demo 58
2.2.18 DMS Java SDK Demo 63
2.3 Python 64
2.3.1 Python SDK Overview 64
2.3.2 Getting Started 64
2.3.3 Usage 66
2.3.4 IAM Python SDK Demo 68
2.3.5 IMS Python SDK Demo 68
2.3.6 VPC Python SDK Demo 70
2.3.7 ECS Python SDK Demo 72
2.3.8 EVS Python SDK Demo 75
2.3.9 RTS Python SDK Demo 76
2.3.10 AS Python SDK Demo 78
2.3.11 Cloud Eye Python SDK Demo 80
2.3.12 DNS Python SDK Demo 86
2.3.13 ELB Python SDK Demo 88
2.3.14 VBS Python SDK Demo 89
2.3.15 CTS Python SDK Demo 93
2.3.16 SMN Python SDK Demo 94
2.3.17 MaaS Python SDK Demo 95
2.3.18 DMS Python SDK Demo 101
2.4 SDK-related FAQ 102
2.4.1 How Can I Create a Key Pair on the Console? 102
2.4.2 How Can I Create a Security Group? 102
2.4.3 How Can I Obtain domain_name, project_name, and project_id? 103
A Mapping Between API and SDK 105
A.1 Java 105
A.1.1 IAM 105
A.1.2 IMS 106
A.1.3 VPC 107
A.1.4 ECS 110
A.1.5 EVS 115
A.1.6 RTS 116
A.1.7 AS 118
A.1.8 CES 121

A.1.9 DNS 122
A.1.10 ELB 123
A.1.11 VBS 125
A.1.12 CTS 126
A.1.13 SMN 127
A.1.14 MaaS 129
A.1.15 DMS 129
A.2 Python 130
A.2.1 IAM 130
A.2.2 IMS 131
A.2.3 VPC 132
A.2.4 ECS 135
A.2.5 EVS 138
A.2.6 RTS 139
A.2.7 AS 140
A.2.8 CES 143
A.2.9 DNS 143
A.2.10 ELB 144
A.2.11 VBS 146
A.2.12 CTS 147
A.2.13 SMN 148
A.2.14 MaaS 150
A.2.15 DMS 150
B OpenStack Client CLI Command 152
C Change History 153

1 Command-Line Interface

1.1 CLI Overview

The command-line interface (CLI) employs the OpenStackClient tool. This tool provides a command line client. You can access cloud services simply by running commands that make calls to the service APIs.

CLI Client Description

The OpenStackClient project is provided with a unified CLI client, so that you can make calls to desired OpenStack project APIs by running commands. Most OpenStack projects have an individual CLI client for each service. For example, the compute service is provided with the Nova CLI client. For details, see https://docs.openstack.org/user-guide/common/cli-overview.html.

Tool Installation

Installation Description

You can install and use OpenStackClient by installing and executing the python-openstackclient plug-in. Therefore, before using the tool, ensure that python-openstackclient is running properly.

This tool can be used in all operating systems (OSs) if Python is running properly. However, the operation methods vary depending on the OS in use. The 64-bit Ubuntu 16.04 OS is recommended. The operations all use 64-bit Ubuntu 16.04 as an example.

You need to install the tool as user **root**.

1. Update the OS.

   Run the following commands to update the OS:

   apt-get update

   apt-get upgrade

2. Install Python.

   Install Python and pip based on the OS in use. Python 2.7 is supported.

   Generally, Ubuntu 16.04 includes Python 2.7. If Python is not installed, perform the following steps to install it:

   Run the following command to install Python:

apt-get install python

Run the following command to install Setuptools:

apt-get install python-setuptools

Run the following command to install pip:

apt-get install python-pip

(If Ubuntu supports Setuptools and pip of earlier versions, you can install them in offline mode.)

Run the following command to install Dev:

apt-get install python-dev

3. Install python-openstackclient and its dependent components.

   The following python-openstackclient versions are supported by default:

   - python-openstackclient: 3.2.1
   - python-novaclient: 6.0.2
   - python-glanceclient: 2.5.0
   - python-keystoneclient: 3.5.1
   - python-neutronclient: 6.0.1
   - python-cinderclient: 1.9.0
   - python-heatclient: 1.5.1
   - python-designateclient: 2.3.0
   - openstacksdk: 0.9.5
   - cliff: 2.2.0
   - os-client-config: 1.21.1
   - osc-lib: 1.1.0
   - Run the following command to install python-openstackclient using pip:

     pip install python-openstackclient==3.2.1

     After the installation is complete, run the following command to verify the installation:

     openstack -h

     Check whether the help information is displayed. The installation is successful if help information is displayed.

     Other components can be installed in the same way in sequence.

Tool Configuration

1. Configure OpenStackClient.

   You can configure the tool either as user **root** or as a common user.

NOTICE

API calls must be made over secure networks, for example, over the VPN or tenants' Elastic Cloud Servers (ECSs), because attacks may be launched over insecure networks.

   a. Switch to the directory where OpenStackClient is installed and create an environment variable file, for example, **novarc**.

b. Use a text editor to edit the environment variable file and fill in the username, password, region, IAM IP address, and port number.

An example is provided as follows:

```
export OS_USERNAME="user_name"
export OS_USER_DOMAIN_NAME=user_domain_name
#export OS_DOMAIN_NAME="domain_name"
export OS_PASSWORD=password
# Only change these for a different region
export OS_TENANT_NAME=tenant_name
export OS_PROJECT_NAME=tenant_name
export OS_AUTH_URL=https://iam.example.com:443/v3
export OS_INTERFACE=public
# No changes needed beyond this point
export NOVA_ENDPOINT_TYPE=publicURL
export OS_ENDPOINT_TYPE=publicURL
export CINDER_ENDPOINT_TYPE=publicURL
export OS_VOLUME_API_VERSION=2
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
```

Table 1-1 lists the environment variables to be configured.

Table 1-1 Environment variables

Parameter	Description
OS_USERNAME	Specifies the username used for running commands. This value is the username for logging in to the management console.
OS_USER_DOMAIN_NAME	Specifies the tenant name. The value is the enterprise account used for logging in to the management console.
OS_DOMAIN_NAME	Specifies the tenant name.
OS_PASSWORD	Specifies the password used for running commands. The value is the password used for logging in to the management console.
OS_TENANT_NAME	Specifies the tenant name used for running commands. The value is the project name displayed on the **Project List** tab on the **My Credential** page.
OS_PROJECT_NAME	Specifies the project name used for running commands. The value is the same as the **OS_TENANT_NAME** value.
OS_AUTH_URL	The parameter value is in the format of https://*IAM URL*:*Port number*/*API version*, for example, https://iam.*example*.com:443/v3. ● *IAM URL*: Obtain the value from Regions and Endpoints.

Parameter	Description
	• *Port number*: 443 • *API* version: v3 (current)
OS_INTERFACE	Specifies the endpoint type. The value of this parameter is **public**.
NOVA_ENDPOINT_TYPE	Specifies the Nova endpoint type. This parameter is mandatory for running OpenStack commands. The value of this parameter is **publicURL**.
OS_ENDPOINT_TYPE	Specifies the OS endpoint type. This parameter is mandatory for running OpenStack commands. The value of this parameter is **publicURL**.
CINDER_ENDPOINT_TYPE	Specifies the Cinder endpoint type. This parameter is mandatory for running OpenStack commands. The value of this parameter is **publicURL**.
OS_VOLUME_API_VERSION	Specifies the Cinder API version. The value of this parameter is **2**.
OS_IDENTITY_API_VERSION	Specifies the authentication API version. The value of this parameter is **3**.
OS_IMAGE_API_VERISON	Specifies the Glance API version. The value of this parameter is **2**.

2. Run the following command to set environment variables:

   source novarc

3. When invoking the Keystone command lines in OpenStackClient, you need to configure **OS_DOMAIN_NAME**, and leave **OS_TENANT_NAME** and **OS_PROJECT_NAME** blank. To do so, run the following commands:

   export OS_DOMAIN_NAME=*domain_name*

   unset OS_TENANT_NAME

   unset OS_PROJECT_NAME

   When invoking commands of other services, you need to leave **OS_DOMAIN_NAME** blank and configure **OS_TENANT_NAME** and **OS_PROJECT_NAME**. To do so, run the following commands:

   unset OS_DOMAIN_NAME

   export OS_TENANT_NAME=*tenant_name*

   export OS_PROJECT_NAME=*project_name*

The CLI becomes available after the tool is installed and configured.

For details about the CLI list supported by the cloud platform, see the appendix.

If you encounter any problem during installation, configuration, or use, resort to the FAQ.

Besides the OpenStack individual CLI and unified CLI, the cloud platform provides the extended CLI through plug-ins. You can install and configure the extended CLI by performing follow-up operations.

(Optional) Installing the Extended CLI

You can use the extended CLI client by running the python-openstackclient plug-in. Therefore, before using the extended CLI client, ensure that python-openstackclient is in the normal state.

Currently, plug-ins of six services are provided. For details, see https://github.com/Huawei?utf8=%E2%9C%93&q=OpenStackClien&type=&language=. Select the service plug-in as needed.

The plug-ins are not submitted to the pip library. Therefore, they cannot be installed using pip. You can download the source plug-in code at the GitHub website (https://github.com/Huawei?utf8=%E2%9C%93&q=OpenStackClien&type=&language=) and then run the **python setup.py install** command to install desired plug-ins. For details, see **Readme** delivered with each plug-in.

(Optional) Configuring the Extended CLI

You need to configure authentication information before using the extended CLI. After the authentication is successful, services will become available to the extended CLI. You can configure the username and password for authentication by configuring environment variables.

Import the following environment variables:

```
export OS_AUTH_URL=<url-to-openstack-identity>
export OS_PROJECT_NAME=<project-name>
export OS_USERNAME=<username>
export OS_PASSWORD=<password>
export OS_REGION_NAME=<region>
# IP address of each service
export OS_ANTIDDOS_ENDPOINT_OVERRIDE=<url-to-endpoint-of-service>
export OS_AS_ENDPOINT_OVERRIDE=<url-to-endpoint-of-service>
export OS_CLOUDEYE_ENDPOINT_OVERRIDE=<url-to-endpoint-of-service>
export OS_VB_ENDPOINT_OVERRIDE=<url-to-endpoint-of-service>
export OS_WORKSPACE_ENDPOINT_OVERRIDE=<url-to-endpoint-of-service>
export OS_KM_ENDPOINT_OVERRIDE=<url-to-endpoint-of-service>
```

After the environment variables are configured, you can use the CLI.

1.2 CLI Use Case

1.2.1 Creating an ECS Using the Individual CLI

Procedure

1. Run the Neutron command to create a port.
2. Run the Cinder command to create a system disk and data disk.
3. Run the Nova command to create an ECS.

Constraints

1. Before creating an ECS, you need to apply for a port and create only one primary network interface card (NIC) for provisioning the ECS.

2. Do not delete the image, network, subnet, and security group when you create an ECS.
3. ECSs cannot be created in batches.
4. If you fail to create an ECS or the created ECS is unavailable, delete it and perform the preceding operations to create another ECS.

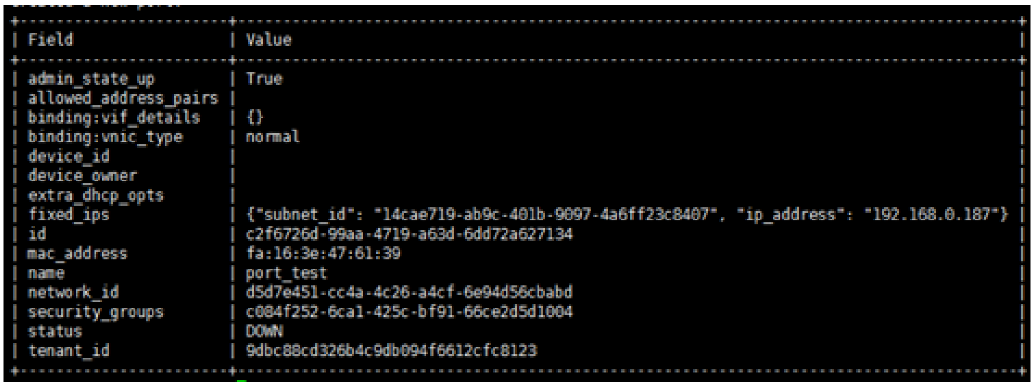
1.2.2 Example of Creating an ECS Using the Individual CLI

Creating a Port

Run the following command to create a port:

neutron port-create d5d7e451-cc4a-4c26-a4cf-6e94d56cbabd --name port_test

Information similar to the following is displayed.



```
+-----------------------+-----------------------------------------------------------------------------------+
| Field                 | Value                                                                             |
+-----------------------+-----------------------------------------------------------------------------------+
| admin_state_up        | True                                                                              |
| allowed_address_pairs |                                                                                   |
| binding:vif_details   | {}                                                                                |
| binding:vnic_type     | normal                                                                            |
| device_id             |                                                                                   |
| device_owner          |                                                                                   |
| extra_dhcp_opts       |                                                                                   |
| fixed_ips             | {"subnet_id": "14cae719-ab9c-401b-9097-4a6ff23c8407", "ip_address": "192.168.0.187"} |
| id                    | c2f6726d-99aa-4719-a63d-6dd72a627134                                              |
| mac_address           | fa:16:3e:47:61:39                                                                 |
| name                  | port_test                                                                         |
| network_id            | d5d7e451-cc4a-4c26-a4cf-6e94d56cbabd                                              |
| security_groups       | c084f252-6ca1-425c-bf91-66ce2d5d1004                                              |
| status                | DOWN                                                                              |
| tenant_id             | 9dbc88cd326b4c9db094f6612cfc8123                                                  |
+-----------------------+-----------------------------------------------------------------------------------+
```

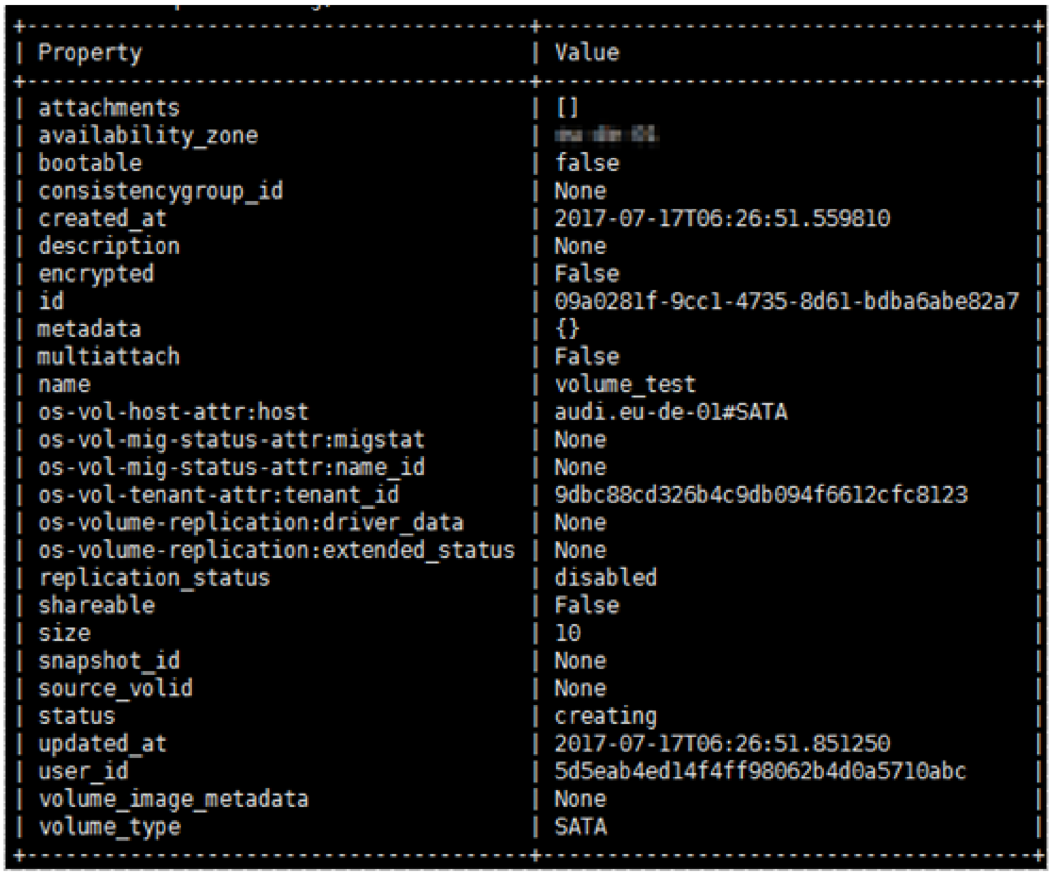
Creating a Mirrored Disk

Run the following command to create a mirrored disk:

cinder create --image-id f6784132-2951-4d1c-82b1-68c540f9fdd6 --volume-type SATA --availability-zone aaa --name volume_test 10

NOTE
In the command, **aaa** indicates the name of an AZ.

Information similar to the following is displayed.



```
+--------------------------------------+--------------------------------------+
| Property                             | Value                                |
+--------------------------------------+--------------------------------------+
| attachments                          | []                                   |
| availability_zone                    | [redacted]                           |
| bootable                             | false                                |
| consistencygroup_id                  | None                                 |
| created_at                           | 2017-07-17T06:26:51.559810           |
| description                          | None                                 |
| encrypted                            | False                                |
| id                                   | 09a0281f-9cc1-4735-8d61-bdba6abe82a7 |
| metadata                             | {}                                   |
| multiattach                          | False                                |
| name                                 | volume_test                          |
| os-vol-host-attr:host                | audi.eu-de-01#SATA                   |
| os-vol-mig-status-attr:migstat       | None                                 |
| os-vol-mig-status-attr:name_id       | None                                 |
| os-vol-tenant-attr:tenant_id         | 9dbc88cd326b4c9db094f6612cfc8123     |
| os-volume-replication:driver_data    | None                                 |
| os-volume-replication:extended_status| None                                 |
| replication_status                   | disabled                             |
| shareable                            | False                                |
| size                                 | 10                                   |
| snapshot_id                          | None                                 |
| source_volid                         | None                                 |
| status                               | creating                             |
| updated_at                           | 2017-07-17T06:26:51.851250           |
| user_id                              | 5d5eab4ed14f4ff98062b4d0a5710abc     |
| volume_image_metadata                | None                                 |
| volume_type                          | SATA                                 |
+--------------------------------------+--------------------------------------+
```

Creating an ECS

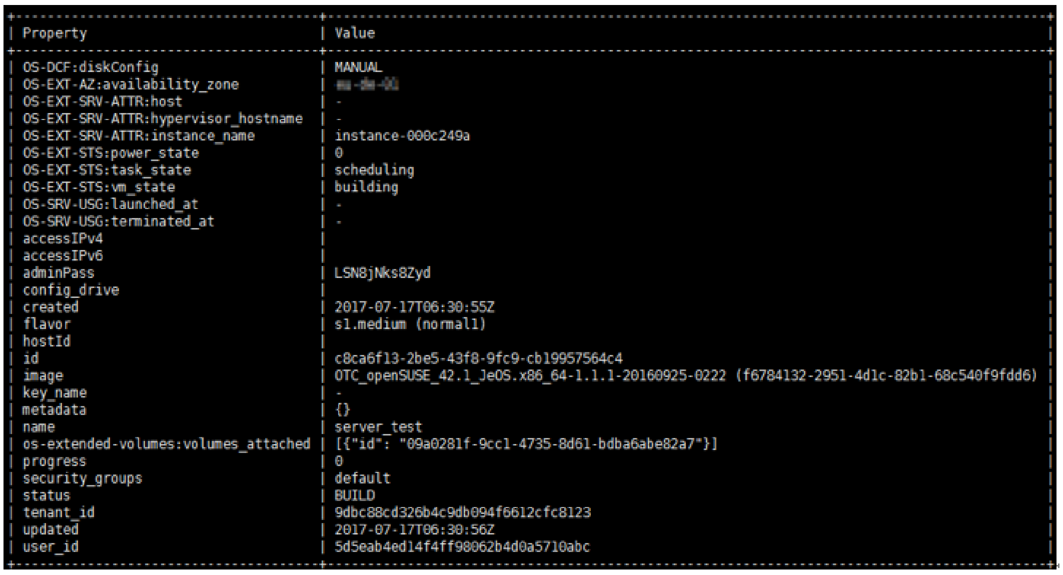
Run the following command to create an ECS:

nova boot --boot-volume 09a0281f-9cc1-4735-8d61-bdba6abe82a7 --availability-zone *aaa* --flavor normal1 --nic port-id=c2f6726d-99aa-4719-a63d-6dd72a627134 server_test

NOTE

In the command, **aaa** indicates the name of an AZ.

Information similar to the following is displayed.



```
+--------------------------------------+------------------------------------------------------------------------------------------+
| Property                             | Value                                                                                    |
+--------------------------------------+------------------------------------------------------------------------------------------+
| OS-DCF:diskConfig                    | MANUAL                                                                                   |
| OS-EXT-AZ:availability_zone          | [redacted]                                                                               |
| OS-EXT-SRV-ATTR:host                 | -                                                                                        |
| OS-EXT-SRV-ATTR:hypervisor_hostname  | -                                                                                        |
| OS-EXT-SRV-ATTR:instance_name        | instance-000c249a                                                                        |
| OS-EXT-STS:power_state               | 0                                                                                        |
| OS-EXT-STS:task_state                | scheduling                                                                               |
| OS-EXT-STS:vm_state                  | building                                                                                 |
| OS-SRV-USG:launched_at               | -                                                                                        |
| OS-SRV-USG:terminated_at             | -                                                                                        |
| accessIPv4                           |                                                                                          |
| accessIPv6                           |                                                                                          |
| adminPass                            | LSN8jNks8Zyd                                                                             |
| config_drive                         |                                                                                          |
| created                              | 2017-07-17T06:30:55Z                                                                     |
| flavor                               | s1.medium (normal1)                                                                      |
| hostId                               |                                                                                          |
| id                                   | c8ca6f13-2be5-43f8-9fc9-cb19957564c4                                                     |
| image                                | OTC_openSUSE_42.1_JeOS.x86_64-1.1.1-20160925-0222 (f6784132-2951-4d1c-82b1-68c540f9fdd6) |
| key_name                             | -                                                                                        |
| metadata                             | {}                                                                                       |
| name                                 | server_test                                                                              |
| os-extended-volumes:volumes_attached | [{"id": "09a0281f-9cc1-4735-8d61-bdba6abe82a7"}]                                         |
| progress                             | 0                                                                                        |
| security_groups                      | default                                                                                  |
| status                               | BUILD                                                                                    |
| tenant_id                            | 9dbc88cd326b4c9db094f6612cfc8123                                                         |
| updated                              | 2017-07-17T06:30:56Z                                                                     |
| user_id                              | 5d5eab4ed14f4ff98062b4d0a5710abc                                                         |
+--------------------------------------+------------------------------------------------------------------------------------------+
```

1.2.3 Creating an ECS Using the Unified CLI

The Unified CLI does not support ECS creation from disks. Instead, you can create an ECS using an image.

Procedure

1. Run the **openstack port create** command to create a port.
2. Run the **openstack server create** command to create an ECS.

Constraints

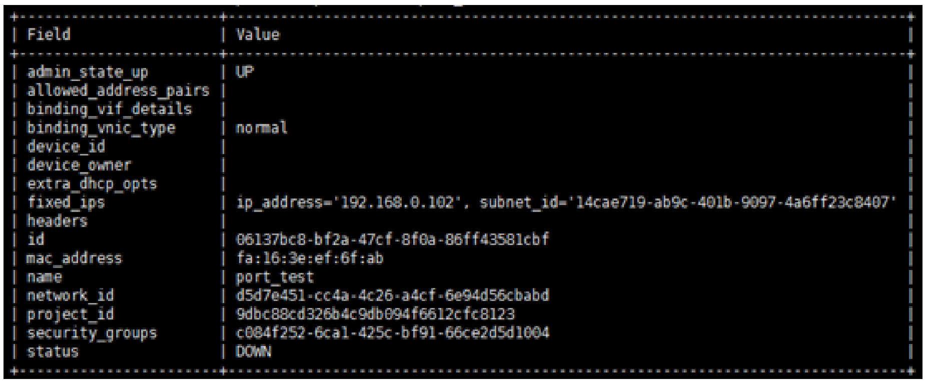
1. Before creating an ECS, you need to apply for a port and create only one primary network interface card (NIC) for provisioning the ECS.
2. Do not delete the image, network, subnet, and security group when you create an ECS.
3. ECSs cannot be created in batches.
4. If you fail to create an ECS or the created ECS is unavailable, delete it and perform the preceding operations to create another ECS.

1.2.4 Use Case for ECS Creation Using Unified CLI

Creating a Port

Run the following command to create a port:

openstack port create port_test --network d5d7e451-cc4a-4c26-a4cf-6e94d56cbabd



```
+-----------------------+-------------------------------------------------------------------------------+
| Field                 | Value                                                                         |
+-----------------------+-------------------------------------------------------------------------------+
| admin_state_up        | UP                                                                            |
| allowed_address_pairs |                                                                               |
| binding_vif_details   |                                                                               |
| binding_vnic_type     | normal                                                                        |
| device_id             |                                                                               |
| device_owner          |                                                                               |
| extra_dhcp_opts       |                                                                               |
| fixed_ips             | ip_address='192.168.0.102', subnet_id='14cae719-ab9c-401b-9097-4a6ff23c8407' |
| headers               |                                                                               |
| id                    | 06137bc8-bf2a-47cf-8f0a-86ff43581cbf                                          |
| mac_address           | fa:16:3e:ef:6f:ab                                                             |
| name                  | port_test                                                                     |
| network_id            | d5d7e451-cc4a-4c26-a4cf-6e94d56cbabd                                          |
| project_id            | 9dbc88cd326b4c9db094f6612cfc8123                                              |
| security_groups       | c084f252-6ca1-425c-bf91-66ce2d5d1004                                          |
| status                | DOWN                                                                          |
+-----------------------+-------------------------------------------------------------------------------+
```

Creating an ECS

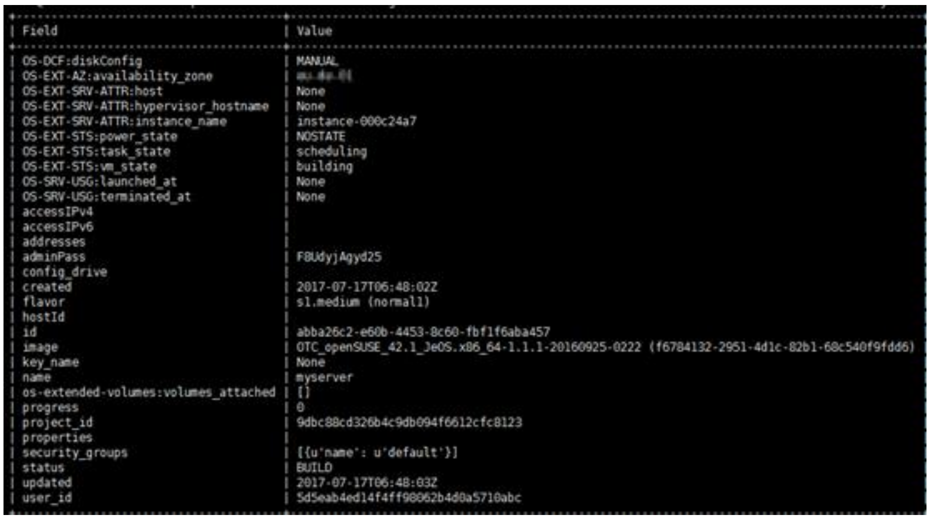
Run the following command to create an ECS:

openstack server create --image f6784132-2951-4d1c-82b1-68c540f9fdd6 --flavor normal1 --availability-zone *aaa* --nic port-id=06137bc8-bf2a-47cf-8f0a-86ff43581cbf myserver

NOTE

In the command, **aaa** indicates the name of an AZ.

The command output is as follows.



```
+--------------------------------------+-----------------------------------------------------------------------------------------+
| Field                                | Value                                                                                   |
+--------------------------------------+-----------------------------------------------------------------------------------------+
| OS-DCF:diskConfig                    | MANUAL                                                                                  |
| OS-EXT-AZ:availability_zone          | [redacted]                                                                              |
| OS-EXT-SRV-ATTR:host                 | None                                                                                    |
| OS-EXT-SRV-ATTR:hypervisor_hostname  | None                                                                                    |
| OS-EXT-SRV-ATTR:instance_name        | instance-000c24a7                                                                       |
| OS-EXT-STS:power_state               | NOSTATE                                                                                 |
| OS-EXT-STS:task_state                | scheduling                                                                              |
| OS-EXT-STS:vm_state                  | building                                                                                |
| OS-SRV-USG:launched_at               | None                                                                                    |
| OS-SRV-USG:terminated_at             | None                                                                                    |
| accessIPv4                           |                                                                                         |
| accessIPv6                           |                                                                                         |
| addresses                            |                                                                                         |
| adminPass                            | F8UdyjAgyd25                                                                            |
| config_drive                         |                                                                                         |
| created                              | 2017-07-17T06:48:02Z                                                                    |
| flavor                               | s1.medium (normal1)                                                                     |
| hostId                               |                                                                                         |
| id                                   | abba26c2-e60b-4453-8c60-fbf1f6aba457                                                    |
| image                                | OTC_openSUSE_42.1_JeOS.x86_64-1.1.1-20160925-0222 (f6784132-2951-4d1c-82b1-68c540f9fdd6) |
| key_name                             | None                                                                                    |
| name                                 | myserver                                                                                |
| os-extended-volumes:volumes_attached | []                                                                                      |
| progress                             | 0                                                                                       |
| project_id                           | 9dbc88cd326b4c9db094f6612cfc8123                                                        |
| properties                           |                                                                                         |
| security_groups                      | [{u'name': u'default'}]                                                                 |
| status                               | BUILD                                                                                   |
| updated                              | 2017-07-17T06:48:03Z                                                                    |
| user_id                              | 5d5eab4ed14f4ff98062b4d0a5710abc                                                        |
+--------------------------------------+-----------------------------------------------------------------------------------------+
```

1.2.5 Use Case for ECS Deletion Using Individual CLI

Run the following command to delete an ECS:

nova delete 3cc81618-4ff4-4e6a-b4df-2a437cd97422

1.2.6 Use Case for ECS Deletion Using Unified CLI

Run the following command to delete an ECS:

openstack server delete 428ac8a5-4fd2-4c22-a33b-f48b46231353

1.3 CLI FAQs

1.3.1 Upgrading Software to the Latest Version

When installing Python, you may need to upgrade software to the latest version.

For example, you can run the following commands to upgrade setuptools and pip:

pip install -U setuptools

pip install -U pip

1.3.2 Rolling Back Software to an Earlier Version

When you install python-openstackclient 3.2.1, its dependent components automatically upgrade to a later version. As a result, some OpenStack commands are unavailable.

If this problem occurs, reinstall the components using the recommended version.

For example, run the following command:

pip install os-client-config==1.21.1

1.3.3 Debugging

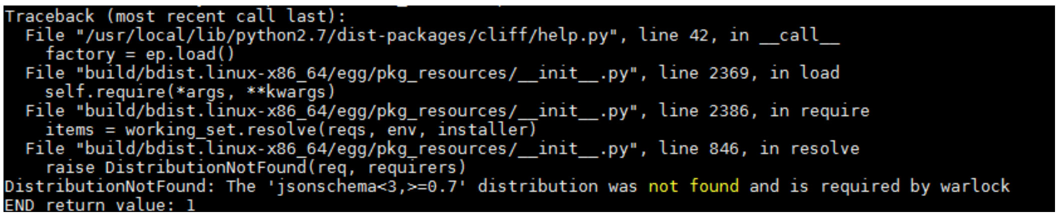
If a fault occurs during the use of the CLI, debugging is a convenient method to locate the fault.

Enabling the debugging function helps quickly locate the fault.

For example, if a problem occurs when you run the **openstack help** command, run the following command to enable debugging:

openstack --debug --help

In the following example, the debugging information indicates that Warlock needs to be installed.



```
Traceback (most recent call last):
  File "/usr/local/lib/python2.7/dist-packages/cliff/help.py", line 42, in __call__
    factory = ep.load()
  File "build/bdist.linux-x86_64/egg/pkg_resources/__init__.py", line 2369, in load
    self.require(*args, **kwargs)
  File "build/bdist.linux-x86_64/egg/pkg_resources/__init__.py", line 2386, in require
    items = working_set.resolve(reqs, env, installer)
  File "build/bdist.linux-x86_64/egg/pkg_resources/__init__.py", line 846, in resolve
    raise DistributionNotFound(req, requirers)
DistributionNotFound: The 'jsonschema<3,>=0.7' distribution was not found and is required by warlock
END return value: 1
```

1.3.4 OpenStack Community CLI Reference

Click here to obtain the latest CLI guide to the OpenStack community.

1.3.5 Cryptography Installation

If Cryptography fails to be installed during the tool installation, run the following command to install build-essential libssl-dev libffi-dev:

apt-get install build-essential libssl-dev libffi-dev

1.3.6 Installing Setuptools and pip Offline

Official Setuptools download website: https://pypi.python.org/pypi/setuptools

Official pip download website: https://pypi.python.org/pypi/pip

Install Setuptools first and then pip.

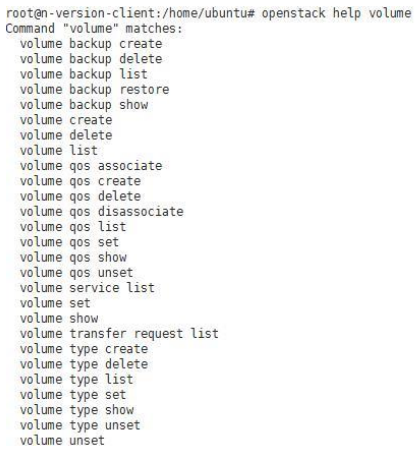
1. Log in to the official websites of Setuptools and pip one by one.
2. Download the installation packages.
3. Upload the packages to the Linux environment.
4. Run the **unzip** or **tar** command to decompress the packages.
5. Go to the decompression directory and run the **python setup.py install** command to complete the installation of Setuptools and pip.

1.3.7 How Can I Obtain Online Help?

You can run the **help** command to obtain online help.

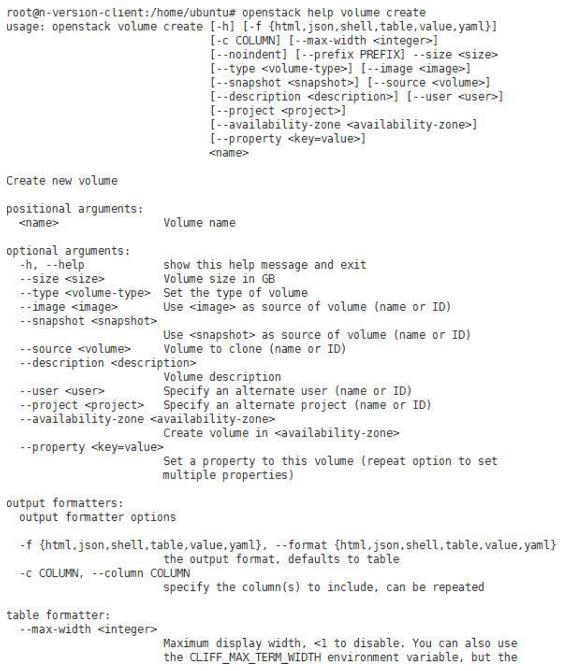
Example:

1. Query required commands.



```
root@n-version-client:/home/ubuntu# openstack help volume
Command "volume" matches:
  volume backup create
  volume backup delete
  volume backup list
  volume backup restore
  volume backup show
  volume create
  volume delete
  volume list
  volume qos associate
  volume qos create
  volume qos delete
  volume qos disassociate
  volume qos list
  volume qos set
  volume qos show
  volume qos unset
  volume service list
  volume set
  volume show
  volume transfer request list
  volume type create
  volume type delete
  volume type list
  volume type set
  volume type show
  volume type unset
  volume unset
```

2. Query related parameters and descriptions.



```
root@n-version-client:/home/ubuntu# openstack help volume create
usage: openstack volume create [-h] [-f {html,json,shell,table,value,yaml}]
                               [-c COLUMN] [--max-width <integer>]
                               [--noindent] [--prefix PREFIX] --size <size>
                               [--type <volume-type>] [--image <image>]
                               [--snapshot <snapshot>] [--source <volume>]
                               [--description <description>] [--user <user>]
                               [--project <project>]
                               [--availability-zone <availability-zone>]
                               [--property <key=value>]
                               <name>

Create new volume

positional arguments:
  <name>                Volume name

optional arguments:
  -h, --help            show this help message and exit
  --size <size>         Volume size in GB
  --type <volume-type>  Set the type of volume
  --image <image>       Use <image> as source of volume (name or ID)
  --snapshot <snapshot>
                        Use <snapshot> as source of volume (name or ID)
  --source <volume>     Volume to clone (name or ID)
  --description <description>
                        Volume description
  --user <user>         Specify an alternate user (name or ID)
  --project <project>   Specify an alternate project (name or ID)
  --availability-zone <availability-zone>
                        Create volume in <availability-zone>
  --property <key=value>
                        Set a property to this volume (repeat option to set
                        multiple properties)

output formatters:
  output formatter options

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

table formatter:
  --max-width <integer>
                        Maximum display width, <1 to disable. You can also use
                        the CLIFF_MAX_TERM_WIDTH environment variable, but the
```

1.3.8 How Can I Obtain the Called APIs?

You can enable the debugging function to obtain the APIs that are called by the executed command.

Example:

nova --debug list

openstack --debug server list

1.3.9 How Can I Obtain CLI Source Code?

The cloud platform supports the OpenStack individual CLI, unified CLI, and extended CLI.

- **Source code for the OpenStack individual CLI:**

  Version: 6.0.2

  https://github.com/openstack/python-novaclient

  Version: 2.5.0

  https://github.com/openstack/python-glanceclient

  Version: 3.5.1

  https://github.com/openstack/python-keystoneclient

  Version: 6.0.1

  https://github.com/openstack/python-neutronclient

  Version: 1.9.0

  https://github.com/openstack/python-cinderclient

  Version: 1.5.1

  https://github.com/openstack/python-heatclient

  Version: 2.3.0

  https://github.com/openstack/python-designateclient

- **Source code for the OpenStack unified CLI:**
  https://github.com/openstack/python-openstackclient/tree/master
  Version: 3.2.1
- **Source code for the extended CLI:**
  https://github.com/Huawei/OpenStackClient_AntiDDOS
  https://github.com/Huawei/OpenStackClient_CES
  https://github.com/Huawei/OpenStackClient_KMS
  https://github.com/Huawei/OpenStackClient_Auto-Scaling
  https://github.com/Huawei/OpenStackClient_VBS
  https://github.com/Huawei/OpenStackClient_Workspace

2 SDK

2.1 SDK Overview

A software development kit (SDK) contains code and examples that you use to create cloud applications in the language of your choice.

Currently, the SDK supports Java and Python languages. If one of the following SDKs do not support your language or examples, you can use the APIs or any other known SDKs.

Java SDK

2.2.2 Getting Started

2.2.5 IMS Java SDK Demo

A.1 Java

Python SDK

2.3.2 Getting Started

2.3.5 IMS Python SDK Demo

A.2 Python

2.2 Java

2.2.1 Java SDK Overview

What Is OpenStack4j?

OpenStack4j is an open source library that helps you manage the OpenStack deployment. The Fluent API it provides helps you fully control various OpenStack services.

The supported Java SDK is developed based on OpenStack4j.

Compatibility Between Java SDK and OpenStack APIs

The following table lists the compatibility between Java SDK and native OpenStack APIs.

OpenStack Component	Cloud Service	API
Keystone	Identity and Access Management (IAM)	V3
Nova	Elastic Cloud Server (ECS)	V2
Neutron	Virtual Private Cloud (VPC)	V2.0
Cinder	Elastic Volume Service (EVS)	V2
Glance	Image Management Service (IMS)	V2

2.2.2 Getting Started

Prerequisites

1. You have obtained required API documents.

   Log in to the following website to obtain the API documents:

   https://support.telefonicaopencloud.com/

   With these documents, you can obtain the OpenStack APIs and related parameters supported by the cloud platform.
2. You have obtained a cloud platform account and provisioned all required services.
3. You have installed JDK. (The Java SDK is applicable to JDK1.7+. You are advised to use JDK1.8.)

SDK Acquisition and Installation

You can download the JAR file from the GitHub website and import the package to the interactive development environment (IDE).

https://github.com/huawei/cloud-sdk-release/tree/master/java-sdk

Take Eclipse as an example. After creating a Java project, perform the following steps to import the JAR file to the new project:

1. Copy the downloaded JAR file to the project folder.
2. Open the project in Eclipse, right-click the project, and choose **Properties**.
3. In the displayed dialog box, click **Java Build Path**. On the **Libraries** tab, click **Add JARs** to add the downloaded JAR file.
4. Click **OK**.

The services involved in this document use the same JAR file.

How to Use

Set parameters, initialize the SDK client, and invoke the SDK to access the service API.

```
package demo;

import java.util.HashMap;
import java.util.List;
import java.util.Map;
import com.huawei.openstack4j.openstack.OSFactory;
import com.huawei.openstack4j.api.OSClient.OSClientV3;
import com.huawei.openstack4j.core.transport.Config;
import com.huawei.openstack4j.model.common.Identifier;
import com.huawei.openstack4j.model.compute.Server;

public class Demo {
public static void main(String[] args) {

// Set the authentication parameters.
String authUrl = "https://iam.example.com/v3"; //endpoint Url
String user = "replace-with-your-username"; // Username
String password = "replace-with-your-password"; //Password
String projectId = "replace-with-your-projectId"; //Project ID
String userDomainId = "replace-with-your-domainId"; //Domain ID

// Initialize the client.
OSClientV3 os = OSFactory.builderV3()
.endpoint(authUrl)
.credentials(user, password, Identifier.byId(userDomainId))
.scopeToProject(Identifier.byId(projectId)).authenticate();

// Set query parameters.
Map<String , String> filter = new HashMap<String, String>();
// Put the parameters that need to be entered into the filter.
filter.put("limit", "3");
// Invoke the interface for querying the VM List.
List<? extends Server> serverList = os.compute().servers().list(filter);
if(serverList.size() > 0) {
System.out.println("get serverList success, size = " + serverList.size());
for (Server server : serverList) {
System.out.println(server);
}
} else {
System.out.println("no server exists.");
}
}
}
```

- **example** in the preceding code is in **Region.Cloud platform domain name** format. For details about the parameters, see here.

Typical Process for Creating an ECS

You can follow the steps below to create an ECS, bind an elastic IP address (EIP) to the ECS, and attach disks to the ECS.

Procedure	Description
1	Connect IAM (Keystone) to AUTH.
2	Obtain your image ID. ● Option 1: Use the public images (recommended). ● Option 2: Use OpenStack Glance v2 to download your images. The

Procedure	Description
	cloud platform supports VHD, ZVHD, QCOW2, and VMDK images.
3	Make a call to the VPC API to obtain the network ID. For details, see Creating a VPC and Subnet.
4	Create an ECS. For details, see Creating an ECS.
5	(Optional) Bind an EIP to the ECS. For details, see Binding an EIP to an ECS.
6	(Optional) Attach disks to ECS. For details, see Attaching a Volume to the ECS.

2.2.3 Usage

Authentication Modes

Java SDK supports two authentication modes: token-based authentication and AK/SK authentication.

For details about the code for token-based authentication, see section 2.2.2 Getting Started.

Sample code for AK/SK authentication

```
package demo;

import java.util.HashMap;
import java.util.List;
import java.util.Map;

import com.huawei.openstack4j.api.OSClient.OSClientAKSK;
import com.huawei.openstack4j.core.transport.Config;
import com.huawei.openstack4j.model.compute.Server;
import com.huawei.openstack4j.openstack.OSFactory;

public class Demo {

 public static void main(String[] args) {

  // Set the authentication parameters.
  String ak = "replace-your-ak";
  String sk = "replace-your-sk";
  String projectId = "replace-your-projectId";
  String region = "replace-your-region";
  String domain = "replace-your-domain";

  OSClientAKSK osclient = OSFactory.builderAKSK().credentials(ak, sk, region,
projectId, domain) .authenticate();

  // Set query parameters.
  Map<String , String> filter = new HashMap<String, String>();
  // Put the parameters that need to be entered into the filter.
  filter.put("limit", "3");
  // Invoke the interface for querying the VM List.
  List<? extends Server> serverList = osclient.compute().servers().list(filter);
  if(serverList.size() > 0)
  {
   System.out.println("get serverList success, size = " + serverList.size());
   for (Server server : serverList) {
    System.out.println(server);
   }
  }
  else {
   System.out.println("no server exists.");
  }
 }
}
```

AK/SK generation description: Log in to the management console, choose **My Credential**, and click **Access Keys** to create an AK and SK.

Service Endpoint Configuration

When using SDK to invoke cloud service APIs, you need to obtain the address (endpoint) of each cloud service.

You can use Java SDK to automatically obtain the endpoints or manually encode the endpoints.

The following are examples of manually encoding endpoints for cloud services:

```
endpointResolver.addOverrideEndpoint(ServiceType.DNS,
 "https://dns.example.com");
endpointResolver.addOverrideEndpoint(ServiceType.VOLUME_BACKUP,
 "https://vbs.example.com/v2/%(project_id)s");
endpointResolver.addOverrideEndpoint(ServiceType.AUTO_SCALING,
 "https://as.example.com/autoscaling-api/v1/%(project_id)s");
endpointResolver.addOverrideEndpoint(ServiceType.CLOUD_EYE,
 "https://ces.example.com/V1.0/%(project_id)s");
endpointResolver.addOverrideEndpoint(ServiceType.LOAD_BALANCER,
 "https://elb.example.com/v1.0/%(project_id)s");
endpointResolver.addOverrideEndpoint(ServiceType.MAP_REDUCE,
 "https://mrs.example.com/v1.1/%(project_id)s");
endpointResolver.addOverrideEndpoint(ServiceType.KEY_MANAGEMENT,
 "https://kms.example.com/v1.0/%(project_id)s");
endpointResolver.addOverrideEndpoint(ServiceType.CLOUD_TRACE,
 "https://cts.example.com/v1.0/%(project_id)s");
endpointResolver.addOverrideEndpoint(ServiceType.ANTI_DDOS,
 "https://antiddos.example.com/v1/%(project_id)s");
endpointResolver.addOverrideEndpoint(ServiceType.Notification,
 "https://smn.example.com/v2/%(project_id)s");
endpointResolver.addOverrideEndpoint(ServiceType.MessageQueue,
 "https://dms.example.com/v1.0/%(project_id)s");
```

- **example** in the preceding code is in **Region.Cloud platform domain name** format. For details about the parameters, see here.
- In the preceding code, you do not need to replace the **project_id** value with the actual value.
- Click here to obtain a complete code example of using Java SDK for reference.

Fault Locating

Add the following code to print the execution details of Java SDK:

```
OSFactory.enableHttpLoggingFilter(true);
```

2.2.4 IAM Java SDK Demo

Service Authentication

IAM is a service that provides API client authentication. After you are authorized by IAM, you can call other service APIs, such as APIs used for creating ECSs.

Authentication code example:

```
OSClientV3 os = OSFactory.builderV3()
.endpoint(("https://iam.example.com/v3")
.credentials("username", "password", Identifier.byName("domain_name"))
.scopeToProject(Identifier.byId("project_id"))
.authenticate();
```

Table 2-1 Parameter description

Parameter	Description
username	Specifies the username.
password	Specifies a password.
domain_name	For details, see 2.4.3 How Can I Obtain domain_name, project_name, and project_id?.
project_id	For details, see 2.4.3 How Can I Obtain domain_name, project_name, and project_id?.

User Management

Domain-level authentication is used for user management operations.

```
OSClientV3 os = OSFactory.builderV3()
.endpoint("https://iam.example.com/v3")
.credentials("username", "password", Identifier.byName("domain_name"))
.scopeToDomain(Identifier.byName("domain_name"))
.authenticate();
```

Table 2-2 Parameter description

Parameter	Description
username	Specifies the username.
password	Specifies the password.
domain_name	Domain name of the user. For details, see 2.4.3 How Can I Obtain domain_name,

Parameter	Description
	project_name, and project_id?.

2.2.5 IMS Java SDK Demo

Public Images

A public image is a widely used and standard image. Each public image contains an OS and multiple pre-installed public applications and is visible to all users. You can configure the OS and software in the public image as needed.

Obtain the image ID from the console as follows:

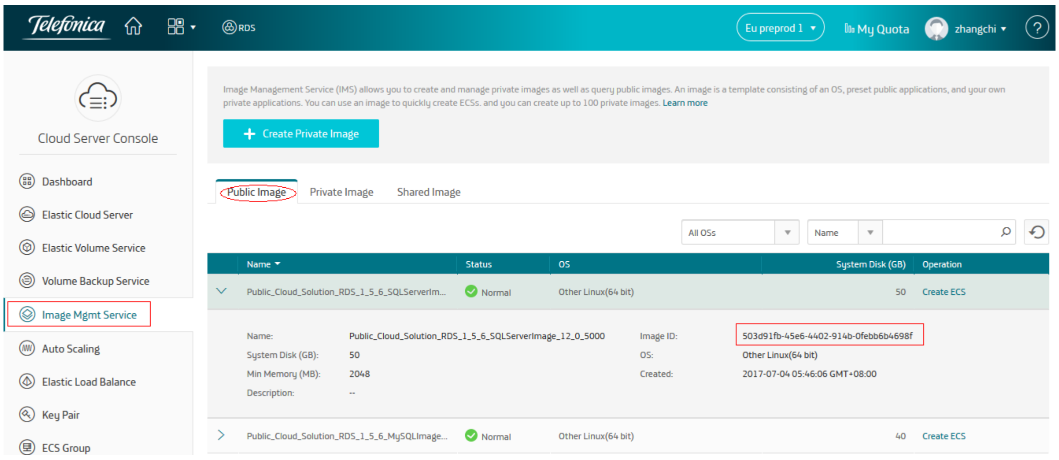


Image Management Service (IMS) allows you to create and manage private images as well as query public images. An image is a template consisting of an OS, preset public applications, and your own private applications. You can use an image to quickly create ECSs. and you can create up to 100 private images. Learn more

	Name	Status	OS	System Disk (GB)	Operation
∨	Public_Cloud_Solution_RDS_1_5_6_SQLServerIm...	✓ Normal	Other Linux(64 bit)	50	Create ECS

Name:	Public_Cloud_Solution_RDS_1_5_6_SQLServerImage_12_0_5000	Image ID:	503d91fb-45e6-4402-914b-0febb6b4698f
System Disk (GB):	50	OS:	Other Linux(64 bit)
Min Memory (MB):	2048	Created:	2017-07-04 05:46:06 GMT+08:00
Description:	--		

	Name	Status	OS	System Disk (GB)	Operation
>	Public_Cloud_Solution_RDS_1_5_6_MySQLImage...	✓ Normal	Other Linux(64 bit)	40	Create ECS

Optionally, you can use the following code to list all the images:

```
os.imagesV2().list()
```

Using OpenStack4j to Create a Private Image

IMS supports the native OpenStack Glance v2 image API, with which you can create a private image with your image file. The supported image types are VHD, ZVHD, QCOW2, and VMDK.

The following uses the image in the QCOW2 format as an example. Image uploading takes a long time, which depends on the image size and network quality.

```
//Create an Image.
Image createdImage = os.imagesV2().create(Builders.imageV2()
.osDistro("ubuntu")
.name("image-name")
.containerFormat(ContainerFormat.BARE)
.visibility(ImageVisibility.PRIVATE)
.diskFormat(DiskFormat.QCOW2)
.architecture("x86_64")
.build());
//Upload the image's image file.
Payload<File> payload = Payloads.create(new File("root.img"));
ActionResponse uploadResult = os.imagesV2().upload(createdImage.getId(), payload,
os.imagesV2().get(createdImage.getId()));
System.out.println("Uploaded done");
```

Table 2-3 Parameter description

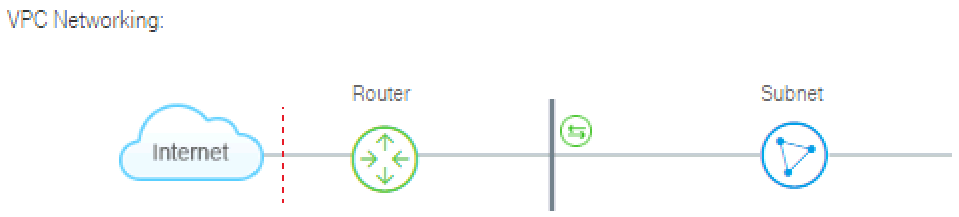
Parameter	Description	Example Value
diskFormat	Specifies the format of the disk.	QCOW2 IMS supports VHD, ZVHD, QCOW2, and VMDK images.

2.2.6 VPC Java SDK Demo

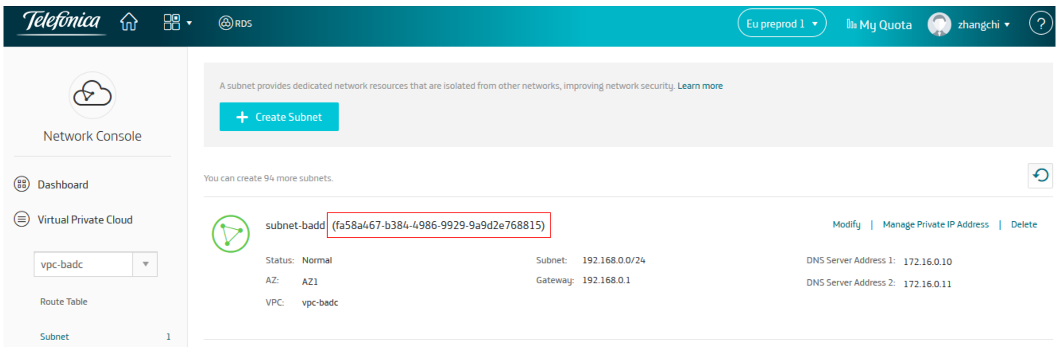
VPC Service OpenStack4j Demo

VPC enables you to provision logically isolated, configurable, and manageable virtual networks for ECSs, improving security of cloud resources and simplifying network deployment.

A typical VPC is composed of a router, network, and subnet, as shown in the following figure.



You can create a VPC on the console and obtain the UUID.



A subnet provides dedicated network resources that are isolated from other networks, improving network security. Learn more

You can create 94 more subnets.

subnet-badd (fa58a467-b384-4986-9929-9a9d2e768815)

Status: Normal
AZ: AZ1
VPC: vpc-badc
Subnet: 192.168.0.0/24
Gateway: 192.168.0.1
DNS Server Address 1: 172.16.0.10
DNS Server Address 2: 172.16.0.11

- Router: A router is a logical entity for forwarding packets across internal subnets and translating the IP addresses of the packets on external networks through an appropriate external gateway.
- Network: A network is an isolated layer-2 network segment, which is similar to a VLAN in the physical network.
- Subnet: A subnet is an IP address segment consisting of IPv4 or IPv6 addresses with their associated configuration status.

Creating a VPC and Subnet

OpenStack4j allows you to create a subnet. The detailed operations are as follows:

1. Create a router.
2. Create a network.
3. Create a subnet.
4. Connect the subnet to the router.

The following code shows the network creation process. You can modify these configurations as required. After you have created the router, network, and subnet and connected the subnet to the router, a new VPC is displayed on the console.

```
public Network createNetwork() {
            //Create a router
            Router router =  os.networking().router().create(Builders.router()
                        .name("routerName")
                        .build());
            //Create a network
            Network network = os.networking().network().create(Builders
                        .network()
                        .name("networkName")
                        .adminStateUp(true)
                        .build());
            //Create a subnet
            Subnet subnet = os.networking().subnet().create(Builders.subnet()
                        .networkId(network.getId())
                        .name("subnetName")
                        .enableDHCP(true)
                        .cidr("192.168.0.0/24")
                        .addDNSNameServer("8.8.8.8")
                        .gateway("192.168.0.1")
                        .build());
            //Connect the subnet to the router, make the router connect to the internet.
            RouterInterface routerinf =
os.networking().router().attachInterface(router.getId(), AttachInterfaceType.SUBNET,
subnet.getId());
            return network;
      }
```

Deleting a VPC

Before deleting a VPC, you need to delete ECSs created in the subnet in the VPC, cancel the association between the router and the subnet, and delete the subnet and the network.

After the ECS deletion command is executed, you can delete the network after ensuring that the ECSs are deleted based on the ECS deletion status. For details about how to delete the network, see the following code:

```
private void clearNet() {
os.networking().router().detachInterface(routerID, subnetID, routerinf.getPortId());
os.networking().subnet().delete(subnetID);
os.networking().network().delete(networkID);
os.networking().router().delete(routerID);
}
```

External Network

An external network is a network with attribute **router:external** set to **true**. This network is used to allocate elastic IP addresses (EIPs). After the EIP is bound to an ECS, the ECS can be accessed from the Internet.

An external network is already available and you do not need to create one.

You can run the **API: GET /v2.0/networks?router:external=True** command to query the ID of the network which is used for creating the EIP.

2.2.7 ECS Java SDK Demo

Creating an ECS

1. Obtain a flavor ID.

   You can run the **os.compute().flavors().list()** command to query all flavors and use a qualified flavor ID to create an ECS.

2. Create a security group.

   For details about how to create a security group, see section 2.4.2 How Can I Create a Security Group?.

   Optionally, you can use OpenStack4j to create a security group based on the following code:

```
org.openstack4j.model.network.SecurityGroup sg =
os.networking().securitygroup().create(Builders.securityGroup()
.name("openstack4j-test-sg")
.build());
SecurityGroupRule rule = Builders.securityGroupRule()
.securityGroupId(sg.getId())
.protocol("tcp")
.direction("ingress")
.portRangeMin(1024)
.portRangeMax(5000)
.build();
SecurityGroupRule sgrule = os.networking().securityrule().create(rule);
```

   Security group rules vary, but need to meet network communication requirements.

3. Create a key pair.

   For details about how to create a private key pair, see section 2.4.1 How Can I Create a Key Pair on the Console?.

   Optionally, you can create a key pair using OpenStack4j based on the following code:

```
String testPublicKey = "ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABAQDdNdUHJHvVaz7ObMKKm4EfqtN8COmcbT7GrCsfrlxz95nUc8G
lAK51VaQjqGU5RwIKPcOfrkJTcXst//pgx7PyzrWrmFlOIjaY8e9HxLQcz2IzrbrbM8TJhB+I3cApdp
wsTqGwjW1xzcMgSrqB0BT7gU4mau0I7Z50RszAhYUVpwGk5OpZGxcXSBQSdSr/KKI6BuMNJYtugGn5d
mr9Ddf99TLbIleLYjmqB0rMjNKHPUxEYMLtixKvXp0qNFfShu7bDp7e2TjhGY8wpda0kC2dDGQJKE18
7N+A3hp2XBZ2UjTcjAVa3C+dwHzHCZd6/yAfJoENe2fDRVnb8cMgm/UX Generated-by-Nova";
Keypair keypair = os.compute().keypairs().create("keyPairName", testPublicKey);
```

4. Create an ECS.

   You can use the following code to create an ECS. You can use interface WaitForServerStatus to continuously query the ECS status until the ECS is in the specified status or the query times out. You can modify the parameters as required. In the following example, the timeout interval is set to 10 minutes.

```
private void createTestServer() throws IOException {
ArrayList networkList = new ArrayList();
networkList.add(net.getId());

ServerCreate serverCreate = Builders.server().name("vm-name")
.flavor("flavorId")
.image("imageId")
.networks(networkList)
.build();
Server server = os.compute().servers().boot(serverCreate);
os.compute().servers().waitForServerStatus(server.getId(), Server.Status.ACTIVE,
10, TimeUnit.MINUTES);
}
```

Table 2-4 Parameter description

Parameter	Description	Example Value
flavorId	Specifies the ID of the flavor.	normal2
imageId	Specifies the ID of the image.	51b2c37f-f5bd-40e0-8aa2-1899a6bbca30

Binding an EIP to an ECS

1. Query the ECS port ID.

   You can query the ECS port ID, which is used for creating the EIP, based on the following code.

```
List<? extends InterfaceAttachment> nicID =
os.compute().servers().interfaces().list(server.getId());
String port_id = nicID.get(0).getPortId();
```

2. Create the EIP.

   You can create the EIP based on the following code.

   As ECS creation takes some time, you need to check the ECS status.

```
NetFloatingIP fip =
os.networking().floatingip().create(Builders.netFloatingIP().floatingNetworkId(
"external_network_id").portId(port_id).build());

if (os.networking().floatingip().get(fip.getId()).getStatus().equals("ACTIVE"))
{
System.out.println("EIP API responsed Success");
}

// judge fip is created successfully
int count = 1;
boolean createFlat = false;
while (count < 10) {
if
(os.networking().floatingip().get(fip.getId()).getStatus().toString().equals("A
CTIVE")) {
System.out.println("Confirmed EIP Create Success");
createFlat = true;
break;
}
count++;
Thread.sleep(1000);
System.out.println(os.networking().floatingip().get(fip.getId()).getStatus());
}
if (!createFlat) {
System.out.println("EIP is not successfully created");
}
```

external_network_id indicates the external network ID. For details, see section External Network.

The EIP is automatically bound to the ECS after being created because you specify the ECS port ID to create the EIP.

Unbinding an EIP from an ECS

You can unbind an EIP from an ECS based on the following code.

The unbinding operation takes several seconds, and the EIP can be deleted only after it is unbound. An error message may be reported if you delete an EIP during the unbinding operation.

```
// after the fip is ACTIVE,Do the remove action
ActionResponse removeAction = os.compute().floatingIps().removeFloatingIP(server,
fip.getFloatingIpAddress());
if (removeAction.isSuccess()) {
System.out.println("Remove Response Action Success");
}

// judge the EIP disassociate successfully
int j = 1;
boolean removeFlag = false;
while (j < 10) {
if (os.networking().floatingip().get(fip.getId()).getPortId() == null
&&
os.networking().floatingip().get(fip.getId()).getStatus().toString().equals("DOWN"
)) {
removeFlag = true;
System.out.println("Confirmed disassociate successfuly");
break;
}
Thread.sleep(1000);
j++;
}
if (!removeFlag) {
System.out.println("Disassociate failure");
}
```

Deleting an ECS

ECS deletion operation consumes some time, and during the process, the port is not deleted immediately.

You need to query the ECS status based on the ECS ID. If **null** is returned, the ECS is deleted, and you can delete the network.

```
ActionResponse deleteRespone = os.compute().servers().delete(server.getId());
System.out.println(os.compute().servers().get(server.getId()));
if (deleteRespone.isSuccess()) {
System.out.println("Delete Server action response :" + deleteRespone);
}

int i = 1;
boolean deleteServerStatus = false;
while (i < 10) {
if (os.compute().servers().get(server.getId()) == null) {
deleteServerStatus = true;
System.out.println("Confirmed delete server");
break;
}
Thread.sleep(1000);
i++;
}

if (!deleteServerStatus) {
System.out.println("Delete Server failed");
}
```

Querying the ECS Status

You can query the ECS status based on the following code:

```
os.compute().servers().get(server_id).getStatus()
```

(Optional) Modifying the ECS Flavor

After the flavor is modified, you can roll back the modification or make the modification take effect.

1. Modify the flavor.
   ```
   os.compute().servers().resize(serverId, flavorId)
   ```
2. Confirm the modification.
   ```
   os.compute().servers().confirmResize(serverId)
   ```
3. Roll back the modification. Rollback cannot be performed if you have already made the modification take effect.
   ```
   os.compute().servers().revertResize(serverId)
   ```

Restarting an ECS

Run the **os.compute().servers().reboot(***serverID*, **RebootType.SOFT)** command to restart an ECS.

Stopping an ECS

Run the **os.compute().servers().action(***serverID***, Action.STOP)** command to stop an ECS.

Image

An image is the OS of a VM and it is a series of files used to create or rebuild servers. By default, carriers provide preset OS images, but you can also create custom images from the ECSs.

List images.

```
import java.util.List;
import com.huawei.openstack4j.model.compute.Image;

List<? extends Image> imgList = os.compute().images().list();
if(imgList.size() > 0) {
 System.out.println("getImageList success, size = " + imgList.size());
}else {
 System.out.println("getImageList failed");
}
```

Obtain details of a specified image.

```
import com.huawei.openstack4j.model.compute.Image;

// get image by id
Image image = os.compute().images().get("imgId");
if(null != image) {
 System.out.println("getImage success, name = " + image.getName());
}else {
 System.out.println("getImage failed");
}
```

Query details of images.

```
import java.util.List;
import com.huawei.openstack4j.model.compute.Image;

List<? extends Image> imgListDetail = os.compute().images().list(true);
if(imgListDetail.size() > 0) {
 System.out.println("getImageListDetail success, size = " + imgListDetail.size());
}else {
 System.out.println("getImageListDetail failed");
}
```

or

```
import java.util.List;
import com.huawei.openstack4j.model.compute.Image;

List<? extends Image> imgListAll = os.compute().images().list(false);
if(imgListAll.size() > 0) {
 System.out.println("getImageListAll success, size = " + imgListAll.size());
}else {
 System.out.println("getImageListAll failed");
}
```

Delete an image.

```
import com.huawei.openstack4j.model.common.ActionResponse;
// delete image by id
ActionResponse rep = os.compute().images().delete("imgId");
```

```
if(rep.isSuccess()) {
 System.out.println("deleteImage success");
}else {
 System.out.println("deleteImage failed");
}
```

Server Group

Create a server group.

```
import com.huawei.openstack4j.model.compute.ServerGroup;

ServerGroup serverGroup = os.compute().serverGroups().create("servergroup_name",
"anti-affinity");
if(null != serverGroup) {
 System.out.println("create serverGroup success, id = " + serverGroup.getId());
}else {
 System.out.println("create serverGroup failed");
}
```

Query details about a specified server group.

```
import com.huawei.openstack4j.model.compute.ServerGroup;

// get ServerGroup by id
ServerGroup serverGroupInfo = os.compute().serverGroups().get("serverGroupId");
if(null != serverGroupInfo) {
 System.out.println("get serverGroupInfo success, name = " +
serverGroupInfo.getName());
}else {
 System.out.println("get serverGroupInfo failed");
}
```

Query the server group list.

```
import com.huawei.openstack4j.model.compute.ServerGroup;
import java.util.List;

List<? extends ServerGroup> list = os.compute().serverGroups().list();
if(list.size() > 0) {
 System.out.println("get ServerGroupList success, size = " + list.size());
}else {
 System.out.println("get ServerGroupList failed");
}
```

Delete a server group.

```
import com.huawei.openstack4j.model.common.ActionResponse;

// delete ServerGroup by id
ActionResponse rep = os.compute().serverGroups().delete("serverGroupId");
if(rep.isSuccess()) {
 System.out.println("deleteserverGroup success");
}else {
 System.out.println("deleteserverGroup failed");
}
```

Attaching or Detaching NICs

Attach a NIC to a server.

```
import com.huawei.openstack4j.model.compute.InterfaceAttachment;

// attach the specified port to the server
InterfaceAttachment newAttach = os.compute().servers().interfaces().create("serId",
"portId");
if(null != newAttach) {
 attachmentId = newAttach.getPortId();
 System.out.println("create InterfaceAttachment success, portStatus = " +
newAttach.getPortState());
}else {
 System.out.println("create InterfaceAttachment failed");
}
```

List server NICs.

```
import com.huawei.openstack4j.model.compute.InterfaceAttachment;
import java.util.List;

// get InterfaceAttachment list by serverId
List<? extends InterfaceAttachment> list =
os.compute().servers().interfaces().list("serId");
if(list.size() > 0) {
 System.out.println("get InterfaceAttachmentList success, size = " + list.size());
}else {
 System.out.println("get InterfaceAttachmentList failed");
}
```

Query details about a specified NIC.

```
import com.huawei.openstack4j.model.compute.InterfaceAttachment;

// get server attachment by attachmentId
InterfaceAttachment gotAttach = os.compute().servers().interfaces().get("serId",
"attachmentId");
if(null != gotAttach) {
 System.out.println("get InterfaceAttachment success, AttachmentId = " +
gotAttach.getPortId());
}else {
 System.out.println("get InterfaceAttachment failed");
}
```

Detach a NIC.

```
import com.huawei.openstack4j.model.common.ActionResponse;

// delete a specified attachmentId
ActionResponse rep = os.compute().servers().interfaces().detach("serId",
"attachmentId");
if(rep.isSuccess()) {
 InterfaceAttachment delAttach = os.compute().servers().interfaces().get("serId",
"attachmentId");
 System.out.println("detach success, deletedAttachment = " + delAttach);
}else {
 System.out.println("detach failed");
}
```

AvailabilityZone

Obtain the AZ list.

```
import java.util.List;
import com.huawei.openstack4j.model.compute.ext.AvailabilityZone;

List<? extends AvailabilityZone> zoneList = os.compute().zones().list();
if(zoneList.size() > 0) {
 System.out.println("get zoneList success, size = " + zoneList.size());
}else {
 System.out.println("get zoneList failed");
}
```

2.2.8 EVS Java SDK Demo

Elastic Volume Service (EVS) disks are scalable virtual block storage devices designed based on the distributed architecture. You can create EVS disks online and attach them to ECSs. The method for using EVS disks is the same as that for using hard disks on physical servers. Compared with traditional hard disks, EVS disks have higher data reliability and I/O throughput capabilities. They are also easier to use. EVS disks apply to file systems, databases, and system software and applications that require block storage devices.

Creating a Volume

You can create a volume using OpenStack4j based on the following code. The volume can be attached to the ECS only when the volume is in the **available** status.

```
Volume volume = os.blockStorage().volumes()
.create(Builders.volume().size(120).name("openstack4j-volume").build());
// wait until volume status available
int createVolumeCount = 1;
boolean createVolumeFlag = false;
while (createVolumeCount < 120) {
if
(os.blockStorage().volumes().get(volume.getId()).getStatus().toString().equals("av
ailable")) {
System.out.println("volume Created successfully");
createVolumeFlag = true;
break;
}
Thread.sleep(1000);
createVolumeCount++;
}
if (!createVolumeFlag) {
System.out.println("Volume created failure");
}
```

Attaching a Volume to the ECS

You can attach the volume to the ECS using OpenStack4j based on the following code. The attachment operation is successful after the volume is in the **in-use** status.

```
//after volume is available, attach to server
VolumeAttachment attachResult = os.compute().servers().attachVolume(server.getId(),
volume.getId(),
"/dev/xvdb");
//adjust if attached successfully
int i = 1;
boolean attachStatus = false;
while (i < 10) {
if
(os.blockStorage().volumes().get(volume.getId()).getStatus().toString().equals("in
-use")) {
attachStatus = true;
System.out.println("volume attached successfully to " +
os.compute().servers().get(server.getId()).getName());
break;
}
Thread.sleep(1000);
i++;
}
if (!attachStatus) {
System.out.println("attach failed");
}
```

Detaching a Volume from the ECS

You can detach the volume from the ECS based on the following code. Detachment execution takes some time, and the volume can be deleted only when the volume is detached.

```
ActionResponse detachResult = os.compute().servers().detachVolume(server.getId(),
attachResult.getId());
System.out.println(detachResult);
assertTrue(detachResult.isSuccess());
int detachCount = 1;
boolean detachStatus = false;
while (detachCount < 60) {
if
(os.blockStorage().volumes().get(volume.getId()).getStatus().toString().equals("av
ailable")) {
detachStatus = true;
System.out.println("volume detached successfully to " );
break;
}
Thread.sleep(1000);
detachCount++;
}
if (!detachStatus) {
System.out.println("detach failed");
}
```

Deleting a Volume

You can delete a volume using OpenStack4j based on the following code.

```
//after detached. delete the volume
ActionResponse deletVolumeResult =
os.blockStorage().volumes().delete(volume.getId());
System.out.println("Delete volume result" + deletVolumeResult);
```

2.2.9 RTS Java SDK Demo

Preparing a Heat Template

A Heat template describes the infrastructure for a cloud application in a text file that is readable and writable by humans.

Prepare the Heat template file, for example, **heatTemplate.yaml**, as shown in the following.

```
---
description: "Simple template to deploy a single compute instance"
heat_template_version: 2013-05-23
parameters:
flavor:
type: string
image:
type: string
key_name:
type: string
network:
type: string
resources:
my_instance:
properties:
availability_zone: az_name
flavor:
get_param: flavor
image:
get_param: image
key_name:
get_param: key_name
networks:
-
network:
get_param: network
type: "OS::Nova::Server"
```

Creating a Stack

The templates enable the creation of most OpenStack resource types, such as instances, EIPs, volumes, security groups, and users. The resources, once created, are referred to as stacks.

You can create a stack based on the following code:

```
public void heatStackTest() throws IOException, InterruptedException{
String testStackName = "test-25";
Map<String, String> stackparams = new HashMap<String, String>();
stackparams.put("network", "network_id");
stackparams.put("flavor", "s1.large");
stackparams.put("image", "imageId");
stackparams.put("key_name", "key_name");

StackCreate stackCreate = Builders.stack()
.name(testStackName)
.templateFromFile("heattemplate.yaml")
.parameters(stackparams)
.build();
Stack stack = os.heat().stacks().create(stackCreate);

//judge the state of the stack
int count = 1;
String getStaus = os.heat().stacks().getDetails(testStackName,
stack.getId()).getStatus().toString();
while ((!getStaus.equals("CREATE_COMPLETE")) && count < 50) {
System.out.println(!getStaus.equals("CREATE_COMPLETE"));
System.out.println(count);
getStaus = os.heat().stacks().getDetails(testStackName,
stack.getId()).getStatus().toString();
System.out.println(getStaus);
Thread.sleep(5000);
count++;
}
}
```

Deleting a Stack

You can delete a stack based on the following code:

```
os.heat().stacks().delete(stackName, stackId);
```

2.2.10 AS Java SDK Demo

Creating an AS Group

An Auto Scaling (AS) group is a set of ECSs with the same application scenario configurations. An AS group defines the minimum and maximum numbers of ECSs.

You can create an AS group using OpenStack4j based on the following code, where the network, security group, and VPC are mandatory parameters. Before creating an AS group, you must create a VPC as well as networks and security groups in this VPC.

```
private String createScalingGroup() {
//network
    IdResourceEntity network = new IdResourceEntity();
    network.setId("fd329aab-d33a-436c-abcf-9ccd4082b2e3");
//securityGroup
    IdResourceEntity securityGroup = new IdResourceEntity();
    securityGroup.setId("57f0a6cd-c427-4e40-a9a2-301ca90893fd");
//az
    String availabilityZone = "eu-de-01";
//group
    ASAutoScalingGroupCreate group =
ASAutoScalingGroupCreate.builder().groupName("test-4-bill")
         .vpcId("0bbc9614-1209-438f-83bb-572b3ad475ea").networks(Lists.newArrayLis
t(network))
         .configId("834b1d4a-36a7-4713-8ec3-41da28f12957").securityGroups(Lists.ne
wArrayList(securityGroup))
         .maxInstanceNumber(2).minInstanceNumber(1).desireInstanceNumber(1).coolDo
wnTime(800)
```

```
         .lbListenerId("4e4f42f1ff004cbdac61f034c9cdfde8")         .lbListenerId("
9ece3b458dd14ce6a15b09073855402e")
         .availabilityZones(Lists.newArrayList(availabilityZone))
         .healthPeriodicAuditMethod(HealthPeriodicAuditMethod.NOVA_AUDIT)
         .healthPeriodicAuditTime(15)
         .instanceTerminatePolicy(InstanceTerminatePolicy.OLD_INSTANCE)
         .deletePublicip(true)
         .build();
//creat group
    ScalingGroupCreate result = osclient.autoScaling().groups().create(group);
    Assert.assertNotNull(result.getGroupId());
    return result.getGroupId();  }
```

Creating an AS Configuration

An AS configuration defines the configurations for creating instances in an AS group. The AS service automatically adds instances to an AS group based on the AS configuration.

You can create an AS configuration based on the following code. When using an existing ECS flavor as the template to create the AS configuration, specify parameter **instance_id**. In this case, parameter **flavorRef**, **imageRef**, and **disks** do not take effect. If **instance_id** is not specified, **flavorRef**, **imageRef**, and **disks** are mandatory.

```
private String createScalingConfig () {
      String keyname = "KeyPair-0406-as";
      Map<String, String> metaData = Maps.newHashMap();
      metaData.put("key1", "val1");
      metaData.put("key2", "val2");
      Disk disk =
Disk.builder().size(40).volumeType(VolumeType.SATA).diskType(DiskType.SYS).build();
//eip
      Bandwidth build =
Bandwidth.builder().chargingMode(Bandwidth.ChargingMode.TRAFFIC).shareType(Bandwid
th.ShareType.PER).size("100").build();
      Eip eip = Eip.builder().ipType("BGP5").bandwidth(build).build();
      PublicIp publicIp = PublicIp.builder().eip(eip).build();
//instanceConfig
      InstanceConfig instanceConfig =
InstanceConfig.builder().instanceId("e926dffb-6fc5-4f8e-b2ff-b2bffe82efb9")
           .flavorRef("c2.medium").imageRef("e215580f-73ad-429d-b6f2-5433947433b0"
).disks(Lists.newArrayList(disk))
           .keyName(keyname)
           .metadata(metaData)
           .publicIp(publicIp)
           .userData("fegrhtht").build();
//createScalingConfig
      ScalingConfigCreate config =
ASAutoScalingConfigCreate.builder().configName("test-config-name")
           .instanceConfig(instanceConfig).build();
      ScalingConfigCreate result = osclient.autoScaling().configs().create(config);
      assertNotNull(result.getConfigId());
      return result.getConfigId();  }
```

Creating an AS Policy

The AS service supports the periodic, scheduled, and alarm policies. If you configure the alarm policy, the selected or created alarm policies can be associated with only one AS group.

You can create periodic, scheduled, and alarm policies based on the following code. The periodic policy can be configured as daily, weekly, or monthly.

```
private void testCreateAutoScalingPolicy() {
    String groupId = "3545d5a1-2d8c-4370-8b95-36f2e8133c24";
//RECURRENCE Daily
   ScheduledPolicy scheduledPolicyDaily =
ScheduledPolicy.builder().launchTime("01:21")
         .recurrenceType(RecurrenceType.DAILY).endTime(getEndTime()).recurrenceVal
ue(null).build();
    ScalingPolicyCreateUpdate policyDaily =
ASAutoScalingPolicyCreateUpdate.builder().policyName("SDK-policyName")
         .groupId(groupId).policyType(ScalingPolicyType.RECURRENCE).scheduledPolic
y(scheduledPolicyDaily).coolDownTime(800)
         .scalingPolicyAction(ScalingPolicyAction.builder().operation(Operation.AD
D).build())
         .build();
    ScalingPolicyCreateUpdate createDaily =
osclient.autoScaling().policies().create(policyDaily);
    assertTrue(createDaily != null
&& !Strings.isNullOrEmpty(createDaily.getPolicyId()));
//RECURRENCE1 Weekly
    ScheduledPolicy scheduledPolicyWeekly =
ScheduledPolicy.builder().launchTime("01:21")
         .recurrenceType(RecurrenceType.WEEKLY).startTime(getStartTime()).endTime(
getEndTime()).recurrenceValue("1,2,3").build();
    ScalingPolicyCreateUpdate policyWeekly =
ASAutoScalingPolicyCreateUpdate.builder().policyName("SDK-policyName")
         .groupId(groupId).policyType(ScalingPolicyType.RECURRENCE).scheduledPolic
y(scheduledPolicyWeekly).coolDownTime(800)
         .scalingPolicyAction(ScalingPolicyAction.builder().operation(Operation.AD
D).instanceNumber(1).build())
         .build();
    ScalingPolicyCreateUpdate createWeekly =
osclient.autoScaling().policies().create(policyWeekly);
    assertTrue(createWeekly != null
&& !Strings.isNullOrEmpty(createWeekly.getPolicyId()));
//RECURRENCE1 Monthly
    ScheduledPolicy scheduledPolicyMonthly =
ScheduledPolicy.builder().launchTime("01:21")
         .recurrenceType(RecurrenceType.MONTHLY).startTime(getStartTime()).endTime
(getEndTime()).recurrenceValue("1,2,3,10").build();
    ScalingPolicyCreateUpdate policyMonthly =
ASAutoScalingPolicyCreateUpdate.builder().policyName("SDK-policyName")
         .groupId(groupId).policyType(ScalingPolicyType.RECURRENCE).scheduledPolic
y(scheduledPolicyMonthly).coolDownTime(800)
         .scalingPolicyAction(ScalingPolicyAction.builder().operation(Operation.AD
D).instanceNumber(1).build())
         .build();
    ScalingPolicyCreateUpdate createMonthly =
osclient.autoScaling().policies().create(policyWeekly0);
```

```
    assertTrue(createMonthly != null
&& !Strings.isNullOrEmpty(createMonthly.getPolicyId()));
//SCHEDULED
     ScheduledPolicy scheduledPolicyScheduled =
ScheduledPolicy.builder().launchTime("2017-07-24T01:21Z").build();
    ScalingPolicyCreateUpdate policyScheduled =
ASAutoScalingPolicyCreateUpdate.builder().policyName("policyTestName")
         .groupId(groupId).policyType(ScalingPolicyType.SCHEDULED).scheduledPolicy
(scheduledPolicy1).coolDownTime(800)
    .scalingPolicyAction(ScalingPolicyAction.builder().operation(Operation.ADD).in
stanceNumber(1).build())
    .build();
    ScalingPolicyCreateUpdate createScheduled =
osclient.autoScaling().policies().create(policyScheduled);
    assertTrue(createScheduled != null
&& !Strings.isNullOrEmpty(createScheduled.getPolicyId()));
//ALARM
     ScalingPolicyCreateUpdate policyAlarm =
ASAutoScalingPolicyCreateUpdate.builder().policyName("policyTestName")
         .groupId(groupId).policyType(ScalingPolicyType.ALARM).alarmId("al14997723
96965q7BBl9MpR").coolDownTime(800)
         .scalingPolicyAction(ScalingPolicyAction.builder().operation(Operation.RE
MOVE).instanceNumber(1).build())
         .build();
    ScalingPolicyCreateUpdate createAlarm =
osclient.autoScaling().policies().create(policyAlarm);
    assertTrue(createAlarm!= null
&& !Strings.isNullOrEmpty(createAlarm.getPolicyId()));  }
```

Creating a Lifecycle Hook

The purpose of adding a lifecycle hook to the AS group is to suspend the instance status to **Wait (Adding to AS group)** or **Wait (Removing from AS group)** during a scaling action. This status retains until the suspension times out or you manually call back the action.

Code reference:

```
import com.huawei.openstack4j.openstack.scaling.domain.ASAutoScalingLifecycleHook;
import
com.huawei.openstack4j.openstack.scaling.domain.ASAutoScalingLifecycleHookType;

//create lifecycle_hook
public ASAutoScalingLifecycleHook creatLifeHook() {

  String groupId = "36f3ac20-e8cf-4c0c-ab4f-78b092b74192";
  String lifecycleHookName = "test-hook";?
  ASAutoScalingLifecycleHookType lifecycleHookType =
ASAutoScalingLifecycleHookType.INSTANCE_LAUNCHING;
  ASAutoScalingDefaultResult defaultResult = ASAutoScalingDefaultResult.ABANDON;
  String notificationTopicUrn =
"urn:smn:cn-suzhou2-1:ebac0c927c104c4587687ce375d0b656:as_test";
  String notificationMetadata = "xxxxxxxx";

  ASAutoScalingLifecycleHook lifecycleHook =
ASAutoScalingLifecycleHook.builder().lifecycleHookName(lifecycleHookName).lifecycl
eHookType(lifecycleHookType.defaultResult(defaultResult).defaultTimeout("86400").n
otificationTopicUrn(notificationTopicUrn).notificationMetadata(notificationMetadat
a).build();

  ASAutoScalingLifecycleHook hook =
osclient.autoScaling().lifecycleHook().create(lifecycleHook, groupId);
  return hook;
 }
```

2.2.11 Cloud Eye Java SDK Demo

Querying Metrics

You can query the metric list in the system and specify the namespace, metric name, dimension, sorting order, start records, and the maximum number of records to filter the search result.

Use the following code to obtain all metrics of the current tenant:

```
//set filter option
MetricFilterOptions config = MetricFilterOptions.create();
MetricFilterOptions options = config.dim(new
String[]{"instance_id,5b4c1602-fb6d-4f1e-87a8-dcf21d9654ba"});
options.limit(50);
options.order(OrderType.ASC);
options.namespace("SYS.ECS");
options.metricName("network_outgoing_bytes_aggregate_rate");
//get some metric
List<? extends Metric> list2 = osclient.cloudEye().metrics().getList(options);
```

Parameter	Description	Example Value
namespace	Specifies the namespace, for example, the ECS namespace.	SYS.ECS
metric_name	Specifies the metric name.	disk_read_bytes_rate
dim	Specifies the metric dimension. A maximum of three dimensions are supported, and the dimensions are numbered from 0 in the **dim.{i}=key,value** format.	AutoScalingGroup,ca3fb7aa-da18-4abc-8206-630cbbb74e14
start	Specifies the paging start value. The format is **namespace.metric_name.key:value**.	SYS.ECS.cpu_util.instance_id:d9112af5-6913-4f3b-bd0a-3f96711e004d
limit	The value ranges from **0** to **1000** (**0** excluded and **1000** included). The default value is **1000**. This parameter is used to limit the number of search results.	50
order	Specifies the sorting order of search results. The value can be **asc** (ascending order) or **desc** (descending order). The default value is **desc**.	desc

Querying the Alarm Rule List

You can query alarm rules and specify the paging parameters to limit the number of search results displayed on a page. You can also set the sorting order of search results.

```
//set filter option
AlarmFilterOptions config = AlarmFilterOptions.create();
AlarmFilterOptions options = config.limit(5);
options.order(OrderType.ASC);
//get some alarm
List<? extends Alarm> list2 = osclient.cloudEye().alarms().getList(options);
```

Parameter	Description	Example Value
start	Specifies the first queried alarm to be displayed on a page. The value is **alarm_id**.	al1498535073312Z27eznaxV
limit	The value ranges from **0** to **100** (**0** excluded and **100** included). The default value is **100**. This parameter is used to limit the number of search results.	50
order	Specifies the sorting order of search results. The value can be **asc** (ascending order) or **desc** (descending order). The default value is **desc**.	desc

Querying an Alarm Rule

You can query the alarm rule based on the alarm ID.

```
//get one alarm
List<? extends Alarm> alarm = osclient.cloudEye().alarms().get(ALARM_ID);
```

Parameter	Description	Example Value
alarm_id	Specifies the alarm rule ID.	al1498535073312Z27eznaxV

Enabling or Disabling an Alarm Rule

You can enable or disable an alarm rule.

```
//start one alarm
ActionResponse actionResponse = osclient.cloudEye().alarms().startAlarm(ALARM_ID);
//stop one alarm
ActionResponse actionResponse = osclient.cloudEye().alarms().stopAlarm(ALARM_ID);
```

Parameter	Description	Example Value
alarm_id	Specifies the alarm rule ID.	al1498535073312Z27eznaxV

Deleting an Alarm Rule

You can delete an alarm rule.

```
//delete one alarm
ActionResponse actionResponse = osclient.cloudEye().alarms().deleteAlarm(ALARM_ID);
```

Parameter	Description	Example Value
alarm_id	Specifies the alarm rule ID.	al1498535073312Z27eznaxV

Querying Monitoring Data

You can query the monitoring data at a specified granularity for a specified metric in a specified period of time. You can specify the dimension of data to be queried.

```
//get one metric data
MetricAggregation metricAggregation = osclient.cloudEye().metricsDatas().get(
"SYS.ECS",
"disk_write_bytes_rate",
new Date(14991341910611),
new Date(14991377910611),
Period.REAL_TIME, Filter.AVERAGE,
new String[]{"instance_id,9191673e-6532-483c-86d7-7514d7dc4d0a"});
```

Parameter	Description	Example Value
namespace	Specifies the namespace, for example, the ECS namespace.	SYS.ECS
metric_name	Specifies the metric name.	disk_read_bytes_rate
from	Specifies the start time of the query. The value is a UNIX timestamp and the unit is ms. Set the value of **from** to at least one period earlier than the current time. Rollup aggregates the raw data generated within a period to the start time of the period. Therefore, if values of **from** and **to** are within a period, the query result will be empty due to the rollup failure. You are advised to set **from** to be at least one period earlier than the current time. Take the 5-minute period as an example. If it is 10:35 now, the raw data generated between 10:30 and 10:35 will be aggregated to 10:30. Therefore, in this example, if the value of **period** is 5 minutes, the value of **from** should be 10:30 or earlier. **NOTE** Cloud Eye rounds up the value of **from** based on the granularity required to perform the rollup.	14991341910611
to	Specifies the end time of the query. The value is a UNIX timestamp and the unit is ms. The value of parameter **from** must be earlier than that of parameter **to**.	14991341892581

Parameter	Description	Example Value
period	Specifies the data monitoring granularity.	Value range: • **1**: The data is monitored in real time. • **300**: The data monitoring granularity is 5 minutes. • **1200**: The data monitoring granularity is 20 minutes. • **3600**: The data monitoring granularity is 1 hour. • **14400**: The data monitoring granularity is 4 hours. • **86400**: The data monitoring granularity is 1 day.
filter	Specifies the data rollup method.	max, min, average, sum, variance
dim	Specifies the metric dimension. A maximum of three dimensions are supported, and the dimensions are numbered from 0 in the **dim.{i}=key,value** format.	AutoScalingGroup,ca3fb7aa-da18-4abc-8206-630cbbb74e14

Adding Monitoring Data

You can add one or multiple pieces of monitoring data.

```
List<CloudEyeMetricData> metrics = new ArrayList<>();
//set dimension
CloudEyeMetricDemension.CloudEyeMetricDemensionBuilder dimBuilder =
CloudEyeMetricDemension.builder().name("instance_id").value("33328f02-3814-422e-b6
88-bfdba93d4050");
CloudEyeMetricDemension dim1 = dimBuilder.build();
List<CloudEyeMetricDemension> dimList = new ArrayList<>();
dimList.add(dim1);
//set namespace, metric name
CloudEyeMetric.CloudEyeMetricBuilder metricBuilder =
CloudEyeMetric.builder().namespace("MINE.APP").metricName("test_add_metric_data_1")
.dimensions(dimList);
//set ttl, collect_time,value,unit
CloudEyeMetricData.CloudEyeMetricDataBuilder builder1 = CloudEyeMetricData.builder()
.metric(metricBuilder.build()).ttl(172800).collectTime(new Date()).value(60)
.unit("%");
```

```
CloudEyeMetric.CloudEyeMetricBuilder metricBuilder2 =
CloudEyeMetric.builder().namespace("MINE.APP").metricName("cpu_util")
.dimensions(dimList);
CloudEyeMetricData.CloudEyeMetricDataBuilder builder2 = CloudEyeMetricData.builder()
.metric(metricBuilder2.build()).ttl(172800).collectTime(new
Date()) .value(70).unit("%");
metrics.add(builder1.build());
metrics.add(builder2.build());
//post metric
ActionResponse actionResponse = osclient.cloudEye().metricsDatas().add(metrics);
```

Parameter	Description	Example Value
metric	Specifies the metric data.	Value in the JSON structure
namespace	Specifies the namespace in the **service.item** format. **service** and **item** each must be a string of 3 to 32 characters starting with a letter and consisting of uppercase letters, lowercase letters, digits, and underscores (_). In addition, **service** cannot be set to **SYS**.	ABC.ECS
metric_name	Specifies the metric name, which must be a string of 1 to 64 characters starting with a letter and consisting of uppercase letters, lowercase letters, digits, and underscores (_).	disk_read_bytes_rate
dimensions	Specifies the list of metric dimensions. Each dimension is a JSON object, and its structure is as follows: **dimension.name**: specifies the dimension name. The value must be a string of 1 to 32 characters starting with a letter and consisting of uppercase letters, lowercase letters, digits, underscores (_), and hyphens (-). **dimension.value**: specifies the dimension value. The value must be a string of 1 to 64 characters starting with a letter and consisting of uppercase letters, lowercase letters, digits, underscores (_), and hyphens (-).	instance_id:33328f02-3814-422e-b688-bfdba93d4050
ttl	Specifies the data validity period. The unit is second. The maximum value is 604,800 seconds. If the validity period expires, the data will be automatically deleted.	172800
collect_time	Specifies the time when the data was collected. The time is UNIX timestamp (ms) format. **NOTE** Since there is a latency between the client and the server, the data timestamp to be inserted should be within the period that starts from three days before	1502938466458

Parameter	Description	Example Value
	the current time plus 20s to 10 minutes after the current time minus 20s. In this way, the timestamp will be inserted to the database without being affected by the latency.	
value	Specifies the metric value.	60
unit	Specifies the data unit.	B
type	Specifies the data type. The value can only be **int** or **float**.	**int** or **float**

Querying Quotas

You can query the total number of resource quotas that can be created and the quota usage. Currently, the resource type can be only the alarm rule.

```
Quota quotas = osclient.cloudEye().quotas().get();
```

2.2.12 DNS Java SDK Demo

Service Description

Domain Name Service (DNS) provides highly available and scalable authoritative DNS resolution services and domain name management services. It translates domain names or application resources into IP addresses required for network connection. By doing so, visitors' access requests are directed to the desired resources.

Creating a Public Zone

You can create a public zone using OpenStack4j based on the following code. After the public zone is created, it will be displayed on the public zone page of the DNS console.

```
public void CreateZones() {
    ZoneBuilder builder = Builders.zone();
    Zone zone =
builder.name(Name).description(Description).email(Email).ttl(TTL).type(zone_type).
build();
    Zone zoneResult = osclient.dns().zones().create(zone);
}
```

Table 2-5 Parameters

Parameter	Description	Example Value
Name	Specifies the domain name registered with the domain name registrar.	example.com
Email	This parameter is optional. Specifies the email address of the administrator managing the public zone. It is recommended that you set the email address	HOSTMASTER@exampl e.com

Parameter	Description	Example Value
	to **HOSTMASTER@*Domain name***.	
Description	This parameter is optional. Provides supplementary information about the zone. The value consists of at most 255 characters.	This is a zone example.
zone_type	Specifies the type of the zone, which can be a public or private zone. ● **public**: specifies the zone that is accessible to hosts on the Internet. ● **private**: specifies the zone that is accessible to the hosts only in the specified VPC. If the parameter is left blank, the system will create a public zone.	public
ttl	Specifies the caching period of the record set (in seconds). The default value is **300**. The value ranges from **300** to **2147483647**.	**5 min** by default

Deleting a Public Zone

You can delete a public zone that you do not need to manage using the DNS service. After the deletion, domain names included in this zone cannot be resolved.

```
public void DeleteZones() {
    Zone deletedZone = osclient.dns().zones().delete(publicZone.getId());
    logger.info("Delete zone: {}", deletedZone);
    if (osclient.dns().zones().get(publicZone.getId()) == null) {
        System.out.println("Confirmed delete zone");
    }
}
```

Creating a Private Zone

To use the DNS service to manage domain names in VPCs, you need to configure private zones on the DNS console. OpenStack4j allows creation of only one private zone. The creation procedure is as follows:

1. Specify the VPC to be associated.
2. Create a private zone.

You can create a private zone using OpenStack4j based on the following code. After the private zone is created, it will be displayed on the private zone page of the DNS console.

```
public void CreatePrivateZones() {
    Router vpv = this.getFirstRouter();
    DesignateZone.Router router = new DesignateZone.Router(vpc.getId(),REGION, Status);
    ZoneBuilder builder = Builders.zone();
    Zone sourceZone =
builder.name(Name).description(Description).email(Email).ttl(TTL).type(zone_type).
router(router).build();
    Zone zoneResult = osclient.dns().zones().create(sourceZone);
logger.info("Create zone: {}", zoneResult);
}
```

Table 2-6 Parameters

Parameter	Description	Example Value
ROUTER_ID	Specifies the VPC to be associated with the private zone.	cd27d47c-ad5a-40a7-8b54-3504a5885d55
REGION	Specifies the region of the VPC to be associated.	eu-de
Status	Specifies the VPC status. The value can be **PENDING_CREATE**, **ACTIVE**, **PENDING_DELETE**, or **ERROR**.	N/A
Name	Specifies the domain name registered with the domain name registrar.	example.com
Email	This parameter is optional. Specifies the email address of the administrator managing the private zone. It is recommended that you set the email address to **HOSTMASTER@*Domain name***.	HOSTMASTER@example.com
Description	This parameter is optional. Provides supplementary information about the zone. The value consists of at most 255 characters.	This a public zone.
zone_type	Specifies the type of the zone, which can be a public or private zone. • **public**: specifies the zone that is accessible to hosts on the Internet. • **private**: specifies the zone that is accessible to the hosts only in the specified VPC. If the parameter is left blank, the system will create a public zone.	private
ttl	Specifies the caching period of the record set (in seconds). The default value is **300**. The value ranges from **300** to **2147483647**.	**5 min** by default

Associating a VPC

You can use OpenStack4j to associate a private zone with a VPC on the cloud platform. The association procedure is as follows:

1. Specify the VPC to be associated.
2. Select the target private zone to associate with the VPC.

```
public void AssociateRouter() {
   router2 = osclient.networking().router().list().get(1);
   DesignateZone.Router router = new DesignateZone.Router(router2.getId(),REGION,
Status);
   DesignateZone.Router routerResult =
osclient.dns().zones().associateRouter(privateZone.getId(),router);
   logger.info("Associate router: {}", routerResult);
}
```

Disassociating a VPC

You can use OpenStack4j to disassociate a private zone from a VPC. The code is as follows:

```
public void DisassociateRouter() {
   getZone();
   DesignateZone.Router router = new
DesignateZone.Router(this.getFirstRouter().getId(), REGION, null);
   DesignateZone.Router routerResult =
osclient.dns().zones().disassociateRouter(privateZone.getId(), router);
   logger.info("Disassociate router: {}", routerResult);
}
```

Deleting a Private Zone

You can delete a private zone that you do not need to manage using the DNS service. After the deletion, domain names included in this zone cannot be resolved.

Before deleting a private zone, ensure that all record sets in this zone have been backed up. The code is as follows:

```
public void DeleteZones() {
   Zone deletedZone = osclient.dns().zones().delete(privateZone.getId());
   logger.info("Delete zone: {}", deletedZone);
   if (osclient.dns().zones().get(privateZone.getId()) == null) {
      System.out.println("Confirmed delete zone");
   }
}
```

2.2.13 ELB Java SDK Demo

Creating a Load Balancer

You can create a load balancer using OpenStack4j based on the following code:

```
public ELBJob create(LoadBalancerCreate loadBalancer) {
   checkArgument(loadBalancer != null, "loadBalancer is required");
checkArgument(!Strings.isNullOrEmpty(loadBalancer.getName()), "name is required");
   checkArgument(!Strings.isNullOrEmpty(loadBalancer.getVpcId()), "vpcId is
required");
   checkArgument(loadBalancer.getType() != null, "type is required");
   checkArgument(loadBalancer.getAdminStateUp() != null, "adminStateUp is required");
   if(Type.INTERNAL.name().equals(loadBalancer.getType()))
{            checkArgument(!Strings.isNullOrEmpty(loadBalancer.getVipSubnetId()),
"vipSubnetId is required when type is Internal");
      checkArgument(!Strings.isNullOrEmpty(loadBalancer.getAzId()), "azId is
required when type is Internal");
      checkArgument(!Strings.isNullOrEmpty(loadBalancer.getTenantId()), "tenantId
is required when type is Internal");
   }
   if(Type.EXTERNAL.name().equals(loadBalancer.getType()))
{       checkArgument(loadBalancer.getBandwidth() != null, "bandwidth is required when
type is External");
   }
   return post(ELBJob.class, uri(API_PATH)).entity(loadBalancer).execute();
}
```

Creating a Listener

You can create a listener using OpenStack4j based on the following code. A listener can be created only when a load balancer is available.

```
public ListenerCreate create(ListenerCreate listener) {
  checkArgument(listener != null, "listener is required");
  checkArgument(!Strings.isNullOrEmpty(listener.getName()), "name is required");
  checkArgument(!Strings.isNullOrEmpty(listener.getLoadBalancerId()),
"loadBalancerId is required");
  checkArgument(listener.getProtocol() != null, "protocol is required");
  checkArgument(listener.getPort() != null, "port is required");
  checkArgument(listener.getBackendProtocol() != null, "backendProtocol is required");
  checkArgument(listener.getBackendPort() != null, "backendPort is required");
  checkArgument(listener.getLbAlgorithm() != null, "lbAlgorithm is required");

  return post(ELBListenerCreate.class, uri(API_PATH)).entity(listener).execute();
}
```

Performing a Health Check

You can perform a health check using OpenStack4j based on the following code. The health check can be performed only when a listener is available.

```
public HealthCheck create(HealthCheckCreate healthCheck) {
  checkArgument(healthCheck != null, "healthCheck is reuquired");
  checkArgument(!Strings.isNullOrEmpty(healthCheck.getListenerId()), "listenerId is
required");

  return post(ELBHealthCheck.class, uri(API_PATH)).entity(healthCheck).execute();
}
```

Adding Members

You can add members to a listener using OpenStack4j based on the following code:

```
public ELBJob create(String listenerId, List<ServerCreate> servers) {
  checkArgument(!Strings.isNullOrEmpty(listenerId), "listenerId is required");
  checkArgument(servers != null && !servers.isEmpty(), "servers is required");
  for (ServerCreate server : servers) {
    checkArgument(server != null, "server can not be null");
    checkArgument(!Strings.isNullOrEmpty(server.getServerId()), "serverId is
required");
    checkArgument(!Strings.isNullOrEmpty(server.getAddress()), "server address is
required");
  }

  return post(ELBJob.class, uri("%s/%s/members", API_PATH,
listenerId)).entity(servers).execute();
}
```

Creating a Certificate

You can create a certificate using OpenStack4j based on the following code:

```
public Certificate create(Certificate cert) {
  checkArgument(cert != null, "cert is required");
  checkArgument(!Strings.isNullOrEmpty(cert.getCertificate()), "certificate is
required");
  checkArgument(!Strings.isNullOrEmpty(cert.getPrivateKey()), "privateKey is
required");

  return post(ELBCertificate.class, uri(API_PATH)).entity(cert).execute();
}
```

2.2.14 VBS Java SDK Demo

Creating a VBS Backup

You can create a VBS backup using OpenStack4j based on the following code. After the VBS backup is created, it will be displayed in the VBS list on the VBS console.

```
public static void createBackup() {
   AsyncVolumeBackupCreate vbc = Builders.asyncVolumeBackupCreate()
          .name(backupName)
          .volumeId(volume.getId())
          .build();
   AsyncVolumeBackupJob job = osclient.blockStorage().asyncBackups().create(vbc);
   Assert.assertNotNull(job.getId());
   backupJobId = job.getId();
}
```

Request parameter description:

Parameter	Mandatory	Type	Description
backup	Yes	dict	Specifies the backup to be created.
volume_id	Yes	string	Specifies the ID of the disk to be backed up.
snapshot_id	No	string	Specifies the snapshot ID of the disk to be backed up.

Parameter	Mandatory	Type	Description
name	Yes	string	Specifies the backup name. The value is a string of 1 to 64 characters consisting of digits, letters, underscores (_), and hyphens (-).
description	No	string	Provides supplementary information about the backup. The value is a string of 1 to 64 characters and cannot contain the less-than sign (<) or greater-than sign (>).

Querying VBS Backup Details

You can query the backup list and obtain the backup details using OpenStack4j based on the following code:

```
public static void queryNativeBackupsDetail(){
    // Without specifying the search criteria
    List<? extends VolumeBackup> list = osclient.blockStorage().backups().list(true);
    Assert.assertNotEquals(list.size(), 0);

    // With the search criteria specified
    HashMap<String, String> filter = new HashMap<>();
    filter.put("name", backupName);
    List<? extends VolumeBackup> list2 = osclient.blockStorage().backups().list(true,
filter);
    for (VolumeBackup backup: list2) {
        Assert.assertEquals(backup.getName(), backupName);
    }
}
```

Request parameter description:

Parameter	Mandatory	Type	Description
name	No	string	Specifies the name of the backup to be queried. This parameter is used to query the backups whose names are specified character strings.
status	No	string	Specifies the status of the backup to be queried. This parameter is used to query the backups in a specified state. The value can be **available**, **error**, **restoring**, **creating**, **deleting**, or **error_deleting**.
offset	No	int	Specifies the offset of the queried details.
limit	No	int	Specifies the maximum number of query results that can be returned.
volume_id	No	string	Specifies the disk ID of the backup to be queried. This parameter is used to query the backups for specific disks.

Restoring a Disk Using a VBS Backup

You can restore a disk from a VBS backup using OpenStack4j based on the following code:

```
public static void restoreBackup() {
   AsyncVolumeBackupJob job = osclient.blockStorage()
         .asyncBackups()
         .restore(backupId, volume.getId());
   Assert.assertNotNull(job.getId());
}
```

Request parameter description:

Parameter	Mandatory	Type	Description
restore	Yes	dict	Specifies the operation of restoring the disk using a backup.
backup_id	Yes	string	Specifies the ID of the backup used to restore a disk.
volume_id	Yes	string	Specifies the ID of the disk to be restored.

Deleting a Backup

You can delete a backup using OpenStack4j based on the following code:

```
public static void deleteNativeBackup()
{
   ActionResponse delete = osclient.blockStorage().backups().delete(backupId);
   Assert.assertEquals(delete.isSuccess(), true);
}
```

Request parameter description:

Parameter	Mandatory	Type	Description
tenant_id	Yes	string	Specifies the ID of the tenant.
backup_id	Yes	string	Specifies the ID of the backup used to restore a disk.

Creating a Backup Policy

You can create a backup policy using OpenStack4j based on the following code:

```
public static void createPolicy()
{
   // Create a scheduled policy first.
```

```
    VBSVolumeBackupScheduledPolicy scheduledPolicy =
VBSVolumeBackupScheduledPolicy.builder()
           .frequency(10)
           .maxBackupAmount(10)
           .retainFirstBackupOfCurrentMonth(true)
           .startTime("01:00")
           .status(VolumeBackupPolicy.VolumeBackupPolicyStatus.OFF)
           .build();
    Assert.assertNotNull(scheduledPolicy);

    // Create a backup policy object.
    VolumeBackupPolicy create = VBSVolumeBackupPolicy.builder()
           .name(policyName)
           .scheduledPolicy(scheduledPolicy)
           .build();
    VolumeBackupPolicy policy = osclient.blockStorage().policies().create(create);
    Assert.assertNotNull(policy.getId());
```

Request parameter description:

Parameter	Mandatory	Type	Description
backup_policy_name	Yes	string	Specifies the backup policy name. The name is a string of 1 to 64 characters consisting of letters, digits, underscores (_), and hyphens (-). It cannot start with **default**.
scheduled_policy	Yes	dict	Specifies details about the scheduling policy.
start_time	Yes	string	Specifies the backup start time, which needs to be converted into the local UTC time (on the hour only). The value is in **HH:mm** format.
frequency	No	integer	Specifies the backup interval (1 to 14 days). Select either this parameter or **week_frequency**. If you select both, this parameter is used.
week_frequency	No	list<dict>	Specifies on which days of each week backup jobs are executed. The value can be one or more of the following: SUN, MON, TUE, WED, THU, FRI, SAT
rentention_num	No	integer	Specifies the retained number (minimum: 2) of backups. Select either this parameter or **rentention_day**. If you select both, this parameter is used.
rentention_day	No	integer	Specifies how many days backups are retained.
remain_first_backup_of_cur	Yes	string	Specifies whether to retain the first backup in

Parameter	Mandatory	Type	Description
Month			the current month. The value can be **Y** or **N**.
status	Yes	string	Specifies the backup policy status. The value can be **ON** or **OFF**.

Deleting a Backup Policy

You can delete a backup policy using OpenStack4j based on the following code:

```
public static void deletePolicy() {
   osclient.blockStorage().policies().delete(policyId);
   List<? extends VolumeBackupPolicy> policies =
osclient.blockStorage().policies().list();
   boolean isSuccess = true;
   for (VolumeBackupPolicy policy:
         policies) {
      if (policy.getId().equals(policyId)) {
         isSuccess = false;
         break;
      }
   }
   Assert.assertEquals(isSuccess, true);
}
```

Request parameter description:

Parameter	Mandatory	Type	Description
tenant_id	Yes	string	Specifies the ID of the tenant.
policy_id	Yes	string	Specifies the ID of the policy.

Querying Backup Policies

You can query backup policies using OpenStack4j based on the following code:

```
public static void queryPolicy() {
   List<? extends VolumeBackupPolicy> policies =
osclient.blockStorage().policies().list();
   boolean isSuccess = false;
   for (VolumeBackupPolicy policy:
        policies) {
      if (policy.getName().equals(policyName)) {
         isSuccess = true;
         policyId = policy.getId();
         break;
      }
   }
   Assert.assertEquals(isSuccess, true);
}
```

2.2.15 CTS Java SDK Demo

Tracker

A tracker will be created after CTS is enabled. All traces recorded by CTS are associated with the tracker.

Creating a Tracker

You can create a tracker using OpenStack4j based on the following code. Currently, only one tracker **system** can be created.

```
public void CreateTracker() {
Tracker create =
osclient.cloudTraceV1().trackers().create(bucket_name,FilePrefixName);
}
```

Deleting a Tracker

You can delete a tracker using OpenStack4j based on the following code:

```
public void DeleteTraker() {
ActionResponse delete = osclient.cloudTraceV1().trackers().delete(tracker_name);
Assert.assertTrue(delete.isSuccess());
List<Tracker> trackers = osclient.cloudTraceV1().trackers().list();
Assert.assertTrue(trackers.size() == 0);
}
```

Updating a Tracker

You can update a tracker using OpenStack4j based on the following code. The information that can be updated includes the bucket name, folder name, status, and the tracker name. The tracker name is optional and can be only **system**.

```
public void UpdateTraker() {
TrackerUpdate update = TrackerUpdate.builder().trackerName(tracker_name)
.bucketName(bucket_name).filePrefixName("SDK-unittest").status(TrackerStatus.Enabl
ed").build();
Tracker updated = osclient.cloudTraceV1().trackers().update(update);
}
```

Querying a Tracker

You can query a tracker using OpenStack4j based on the following code by specifying **tracker_name**:

```
public void GetTraker() {
Tracker get = osclient.cloudTraceV1().trackers().get(tracker_name);
}
```

Trace

This interface is used to query records of operations on resources during the last seven days.

Querying the Trace List

You can query a trace list using OpenStack4j based on the following code. You can filter out required traces by specifying multiple parameters.

```
/*v1 interface*/
/*v2 interface*/
public void ListTrace() {
TraceListOptions options =
TraceListOptions.create().limit(5).user("renxiaomei").serviceType("CTS");
List<Trace> list = osclient.cloudTraceV2().traces().list("system", options);
if (list.size() > 0) {
Trace trace = list.get(list.size() - 1);
options.marker(trace.getId());
List<Trace> list2 = osclient.cloudTraceV2().traces().list("system", options);
}
```

2.2.16 SMN Java SDK Demo

Creating a Topic

You can create a maximum of 3000 topics. APIs are idempotent. If a topic of the same name already exists, the status code 200 is returned. Otherwise, the status code 201 is returned.

You can create a topic using OpenStack4j based on the following code:

```
Topic topic = osclient.notification().topics().create("topic-name", "display-name");
```

Adding a Subscription

Add a subscription to a specified topic. If the status of the subscription is unconfirmed, a confirmation message is sent to the subscriber. After confirming the subscription, the subscriber can receive notification messages published to the topic. APIs are idempotent. If the added subscription already exists, the status code 200 is returned. Otherwise, the status code 201 is returned.

You can add a subscription using OpenStack4j based on the following code by specifying **tp** to **topicUrn**:

```
SubscriptionCreate subscribe =
SubscriptionCreate.builder().topicUrn("topic-urn").endpoint("xx@xx.com") .protocol
(Protocol.EMAIL).remark("sdk-unittest").build();
Subscription subscription =
osclient.notification().subscriptions().subscribe(subscribe);
```

Publishing a Message

You can publish messages to a topic. After the message ID is returned, the message has been saved and is to be pushed to the subscribers of the topic. The message format varies depending on the protocol of a subscription.

You can publish a message using OpenStack4j based on the following code:

```
MessageIdResponse message = osclient.notification().messages().publish("topic-urn",
"subject", "message-content");
```

2.2.17 MaaS Java SDK Demo

Object Storage Migration Service (MaaSOBS) online migrates OBS of another cloud service provider to the destination cloud platform. This migration service offers secure object authentication, encrypted data transmission, and reliable interruption recovery.

Obtaining the Service Version

You can obtain the current MaaS service version using OpenStack4j based on the following code:

```
Version[] version = osclient.maas().version().get();
```

Creating a Migration Task

You can create a migration task using OpenStack4j based on the following code. After the migration task is created, it is added to the task queue and waits for execution.

```
Node srcNode = Node.builder().region(srcRegion).ak(srcAk).sk(srcSk)
.objectKey(srcObjectKey).bucket(srcBucket).build();
Node dstNode = Node.builder().region(dstRegion).ak(dstAk).sk(dstSk)
.objectKey(dstObjectKey).bucket(dstBucket).build();
TaskCreateBuilder taskBuilder =
TaskCreate.builder().srcNode(srcNode).dstNode(dstNode).enableKMS(isKMS).threadNum(
threadNum).description("description");
TaskCreate create = taskBuilder.build();
TaskCreateResp resp = osclient.maas().task().create(create);
taskId = resp.getId();
```

Table 2-7 Parameter description

Parameter	Description	Example Value
srcRegion	Specifies the source region ID. Note that this parameter is not the region name.	us-east-1
srcAk	Specifies the access key ID on the source end. In most cases, the value is a string of 20 characters consisting of digits and letters.	AKTAI72RCUBV4AHQO46C
srcSk	Specifies the secret access key ID on the source end. In most cases, the value is a string of 40 characters consisting of digits and letters.	32o6vpFgj76zII1HOad7SrbygHChx9TbwWpDzsHo
srcObjectKey	Specifies the JSON character string of the migration object on the	{ path: "test-01/", keys:

Parameter	Description	Example Value
	source end. In most cases, the value is in the path+keys array. **path** indicates the parent path of the object on the source end. **keys** indicates the object array in the path. If the migration object is a folder, the value ends with a slash (/).	["10000-files/","rmb001"] }
srcBucket	Specifies the bucket name on the source end. In most cases, the value is the name of a bucket created by the provider of the public cloud supporting service migration.	Filp-srouce-bucket
dstRegion	Specifies the region ID of OBS on the destination end. Note that this parameter is not the region name.	test-region-1
dstAk	Specifies the access key ID on the destination end. In most cases, the value is a string of 20 characters consisting of digits and letters.	KEIIFOFUCVDQJ0E0NW4S
dstSk	Specifies the secret access key ID on the destination end. In most cases, the value is a string of 40 characters consisting of digits and letters.	32o6vpOg376zII1HOad7SrbyfHChx9TbwwprzsHN
dstObjectKey	Specifies the path on the destination end. The value ends with a slash (/).	"test-dir/"
dstBucket	Specifies the bucket name on the destination end. In most cases, the value is the name of a bucket created in the region of OBS.	Bucket-maasobs
isKMS	Specifies whether to enable the KMS function to encrypt the object data after the migration. The value is a boolean type.	True/False

Parameter	Description	Example Value
threadNum	Specifies the number of threads used for migration. The default value is **5**.	5
description	Provides supplementary information about the migration task. This parameter is optional.	"this is test"

Pausing a Migration Task

You can pause a migration task using OpenStack4j based on the following code by specifying the task ID:

```
waitTaskToState(State.EXECUTE);
ActionResponse resp = osclient.maas().task().stop(taskId);
```

Table 2-8 Parameter description

Parameter	Description	Example Value
taskId	Specifies the ID of the migration task to be paused. The value is a string of digits.	172315251263

Continuing a Migration Task

You can continue a paused or failed migration task using OpenStack4j based on the following code by specifying required parameters, including the respective AKs and SKs on the source end and destination end:

```
waitTaskToState(State.STOP);
TaskStart task = TaskStart.builder().sourceAk(srcAk).sourceSk(srcSk)
.targetAk(dstAk).targetSk(dstSk).build();
ActionResponse resp = osclient.maas().task().start(taskId, task);
```

Table 2-9 Parameter description

Parameter	Description	Example Value
srcAk	Specifies the access key ID on the source end for the migration task. In most cases, the value is a string of 20 characters consisting of digits and letters.	AKTAI72RCUBV4AHQO46C
srcSk	Specifies the secret access	32o6vpFgj76zII1HOad7Srb

Parameter	Description	Example Value
	key ID on the source end for the migration task. In most cases, the value is a string of 40 characters consisting of digits and letters.	ygHChx9TbwWpDzsHo
dstAk	Specifies the access key ID on the destination end for the migration task. In most cases, the value is a string of 20 characters consisting of digits and letters.	KEIIFOFUCVDQJ0E0NW4S
dstSk	Specifies the secret access key ID on the destination end for the migration task. In most cases, the value is a string of 40 characters consisting of digits and letters.	32o6vpOg376zII1HOad7SrbyfHChx9TbwwprzsHN

Deleting a Migration Task

You can delete an executed or waiting migration task using OpenStack4j based on the following code by specifying the task ID:

```
waitTaskToState(State.STOP);
ActionResponse resp = osclient.maas().task().delete(taskId);
```

Table 2-10 Parameter description

Parameter	Description	Example Value
taskId	Specifies the ID of the migration task to be deleted. The value is a string of digits.	172315251263

Querying Details of a Migration Task

You can query details of a migration task using OpenStack4j based on the following code by specifying the task ID:

```
Task task = osclient.maas().task().get(taskId);
```

Table 2-11 Parameter description

Parameter	Description	Example Value

Parameter	Description	Example Value
taskId	Specifies the ID of the migration task to be queried. The value is a string of digits.	172315251263

Querying All Tasks of a Tenant

You can query details of a specified number of migration tasks using OpenStack4j based on the following code by specifying three parameters. The first parameter specifies the start number of the migration tasks to be queried, the second parameter specifies the total number of migration tasks to be queried, and the third parameter specifies the status of the migration tasks to be queried. If the task status is not specified, migration tasks of all states will be queried.

```
TaskListOptions options =
TaskListOptions.create().start(startNum).limit(limitCount).state(Status);
Task[] list = osclient.maas().task().list(options);
```

Table 2-12 Parameter description

Parameter	Description	Example Value
startNum	Specifies the start serial number.	1
limitCount	Specifies the maximum number of returned tasks, which cannot exceed 100. Otherwise, the query fails.	5
Status	Specifies the status of the tasks to be queried. If this parameter is left blank, the tasks of all states will be queried.	State.***SUCCESS*** State.***EXECUTE*** State.***FAILED*** State.***STOP*** State.***WAIT***

Querying the Total Number of Migration Tasks

You can query either the total number of migration tasks in a specified state or the total number of migration tasks in all states using OpenStack4j based on the following code.

Querying the total number of migration tasks in all states:

```
long count = osclient.maas().task().count();
```

Querying the total number of migration tasks in a specified state:

```
long count = osclient.maas().task().count(Status);
```

Table 2-13 Parameter description

Parameter	Description	Example Value
Status	Specifies the status of the tasks to be queried. If this parameter is left blank, the tasks of all states will be queried.	State.***SUCCESS*** State.***EXECUTE*** State.***FAILED*** State.***STOP*** State.***WAIT***

2.2.18 DMS Java SDK Demo

Distributed Message Service (DMS) is a message middleware service based on distributed, high-availability clustering technology. It provides reliable, scalable, fully managed queues for storing messages. DMS enables cloud applications to decouple from each other, achieving high cost-effectiveness.

Creating a Queue

You can create a queue using OpenStack4j based on the following code. After the queue is created, messages will be sent to this queue.

```
String name = randomName();
String description = "sdk-unittest"
Queue queue = null;
queue = osclient.messageQueue().queue().create(name, description);
```

Creating a Consumer Group

You can create a consumer group using OpenStack4j based on the following code. After the consumer group is created, it can consume messages in the queue.

```
List<ConsumerGroup> groups = null;
List<String> groupNames = Lists.newArrayList("consumer-group-1", "consumer-group-2");
queueId queueID = queue.getId();
groups = osclient.messageQueue().consumerGroups().create(queueID, groupNames);
```

Producing Messages

You can produce messages using OpenStack4j based on the following code:

```
public void testProduceMessage() {
HashMap<String, Object> attributes1 = Maps.newHashMap();
attributes1.put("attr1", 1);
attributes1.put("attr2", false);
QueueMessage message =
QueueMessage.builder().body("sdk-unittests").attributes(attributes1).build();
ActionResponse produce = osclient.messageQueue().messages().produce(queue.getId(),
message);
}
```

Consume Messages

You can consume messages using OpenStack4j based on the following code:

```
public void testConsumeMessages() {
ConsumerGroup consumerGroup1 = groups.get(0);
List<QueueMessageWithHandler> all = Lists.newArrayList();
for (int i = 0; i < 3; i++) {
List<QueueMessageWithHandler> temp =
osclient.messageQueue().messages().consume(queue.getId(), consumerGroup1.getId(), 5,
10);
all.addAll(temp);
}
}
```

2.3 Python

2.3.1 Python SDK Overview

What Is Python OpenStack SDK?

Python OpenStack SDK is a collection of a LIB library that creates applications on OpenStack-based cloud platforms. It targets at helping users to interact with OpenStack services and provides comprehensive documents, examples, and tools.

The supported Python SDK is developed based on the Python OpenStack SDK.

Compatibility Between Python SDK and OpenStack APIs

The following table lists the compatibility between Python SDK and native OpenStack APIs.

OpenStack Component	Cloud Service	API
Keystone	IAM	V3
Nova	ECS	V2
Neutron	VPC	V2.0
Cinder	EVS	V2
Glance	IMS	V2

2.3.2 Getting Started

Prerequisites

1. You have obtained required API documents.
   Log in to the following website to obtain the API documents:
   https://support.telefonicaopencloud.com/

With these documents, you can obtain the OpenStack APIs and related parameters supported by the cloud platform.

2. You have obtained a cloud platform account and provisioned all required services.
3. You have installed Python, pip, and git. Python SDK is applicable to Python 2.7.x.

SDK Acquisition and Installation

You can download the source code from the GitHub website to install the SDK.

Download the code to a proper position of your project. Use *pythonsdk* as an example. Run the following commands to download the source code and install the SDK:

git clone https://github.com/huawei/cloud-sdk-python *pythonsdk*

cd *pythonsdk*

pip install -r requirements.txt

python setup.py install

How to Use

Set parameters, create connections, and invoke the SDK to access the service API.

```
from openstack import connection

# create connection
username = "replace-with-your-username"
password = "replace-with-your-password"
projectId = "replace-with-your-projectId"    # tenant ID
userDomainId = "replace-with-your-domainId"    # user account ID
auth_url = " https://iam.example.com/v3"    # endpoint url
conn = connection.Connection(auth_url=auth_url,
                        user_domain_id=userDomainId,
                        project_id=projectId,
                        username=username,
                        password=password)

# set parameters
limit = 5

# define function for listing servers
def list_servers():
   # get server list with params
   servers = conn.compute.servers(limit=limit)
   # iterate servers list
   for server in servers:
      print(server)

# visit API
list_servers()
```

- **example** in the preceding code is in **Region.Cloud platform domain name** format. For details about the parameters, see here.

2.3.3 Usage

Configuration on the Client

Some functions supported by SDK can be enabled or disabled through configuration.

Sample code

```
conn = connection.Connection(auth_url=auth_url,
                             user_domain_id=userDomainId,
                             project_id=projectId,
                             username=username,
                             password=password,
                             verify=False)
```

Currently, the following custom parameters are supported.

Parameter	Default Value	Function Description	Remarks
verify	True	SSL check	You are advised to set **verify** to **True**.

Service Endpoint Configuration

When using SDK to invoke cloud service APIs, you need to obtain the address (endpoint) of each cloud service.

You can use Python SDK to automatically obtain the endpoints or manually encode the endpoints.

The following are examples of manually encoding endpoints for cloud services:

```
os.environ.setdefault(
    'OS_CLOUD_EYE_ENDPOINT_OVERRIDE',
    'https://ces.example.com/V1.0/%(project_id)s'
)
os.environ.setdefault(
    'OS_AUTO_SCALING_ENDPOINT_OVERRIDE',
    ('https://as.example.com'
     '/autoscaling-api/v1/%(project_id)s')
)
os.environ.setdefault(
    'OS_DNS_ENDPOINT_OVERRIDE',
    'https://dns.example.com/v2'
)
os.environ.setdefault(
    'OS_VOLUME_BACKUP_ENDPOINT_OVERRIDE',
    'https://vbs.example.com/v2/%(project_id)s'
)
os.environ.setdefault(
    'OS_LOAD_BALANCER_ENDPOINT_OVERRIDE',
    'https://elb.example.com/v1.0/%(project_id)s'
)
```

```
os.environ.setdefault(
    'OS_MAP_REDUCE_ENDPOINT_OVERRIDE',
    'https://mrs.example.com/v1.1/%(project_id)s'
)
os.environ.setdefault(
    'OS_CTS_ENDPOINT_OVERRIDE',
    'https://cts.example.com/v1.0/%(project_id)s'
)
os.environ.setdefault(
    'OS_SMN_ENDPOINT_OVERRIDE',
    'https://smn.example.com/v2/%(project_id)s'
)
os.environ.setdefault(
    'OS_MAAS_ENDPOINT_OVERRIDE',
    'https://maas.example.com/v1/%(project_id)s'
)
os.environ.setdefault(
    'OS_KMS_ENDPOINT_OVERRIDE',
    'https://kms.example.com/v1.0/%(project_id)s'
)
os.environ.setdefault(
    'OS_ANTI_DDOS_ENDPOINT_OVERRIDE',
    'https://antiddos.example.com/v1/%(project_id)s'
)
os.environ.setdefault(
    'OS_DMS_ENDPOINT_OVERRIDE',
    'https://dms.example.com/v1.0/%(project_id)s'
)
os.environ.setdefault(
    'OS_RDSV1_ENDPOINT_OVERRIDE',
    'https://rds.example.com/rds/v1/%(project_id)s'
)
os.environ.setdefault(
    'OS_CDN_ENDPOINT_OVERRIDE',
    'https://cdn.example.com/v1.0'
)
```

- **example** in the preceding code is in **Region.Cloud platform domain name** format. For details about the parameters, see here.
- In the preceding code, you do not need to replace the **project_id** value with the actual value.
- Click here to obtain a complete code example of using Python SDK for reference.

Fault Locating

To enable the debugging function using Python SDK, add the following code to the application:

```
from openstack import utils
utils.enable_logging(debug=True,stream=sys.stdout)
```

2.3.4 IAM Python SDK Demo

Service Authentication

IAM is a service that provides API client authentication. After you are authorized by IAM, you can call other service APIs, such as APIs used for creating ECSs.

After being authenticated, you can manage the IAM, ECS, EVS, VPC, and RTS services.

Example authentication code:

```
def create_connection(auth_url, region, project_name, username, password,
user_domain_name):
    return connection.Connection(
        'auth_url': auth_url,
        'project_name': project_name,
        'username': username,
        'password': pasword,
        "region": region,
        "user_domain_name":user_domain_name
        )
```

The following table lists parameters descriptions.

Table 2-14 Parameter description

Parameter	Description
auth_url	Specifies the IAM authentication URL.
username	Specifies the username.
user_domain_name	Specifies the tenant name.
password	Specifies the password.
project_name	For details, see 2.4.3 How Can I Obtain domain_name, project_name, and project_id?.

2.3.5 IMS Python SDK Demo

Public Images

A public image is a widely used and standard image. Each public image contains an OS and multiple pre-installed public applications and is visible to all users.

Obtain the image ID from the console as follows:

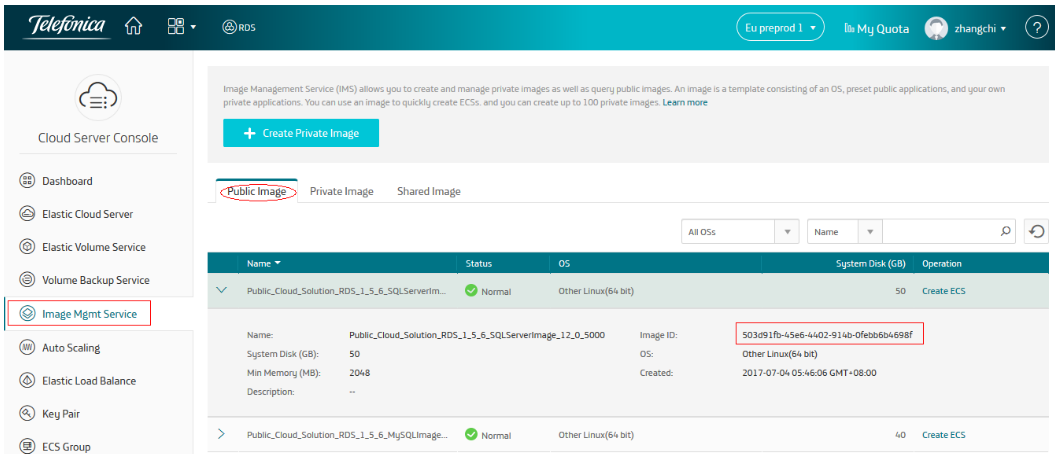


Image Management Service (IMS) allows you to create and manage private images as well as query public images. An image is a template consisting of an OS, preset public applications, and your own private applications. You can use an image to quickly create ECSs. and you can create up to 100 private images. Learn more

Name	Status	OS	System Disk (GB)	Operation
Public_Cloud_Solution_RDS_1_5_6_SQLServerIm...	Normal	Other Linux(64 bit)	50	Create ECS
Public_Cloud_Solution_RDS_1_5_6_MySQLImage...	Normal	Other Linux(64 bit)	40	Create ECS

Name:	Public_Cloud_Solution_RDS_1_5_6_SQLServerImage_12_0_5000	Image ID:	503d91fb-45e6-4402-914b-0febb6b4698f
System Disk (GB):	50	OS:	Other Linux(64 bit)
Min Memory (MB):	2048	Created:	2017-07-04 05:46:06 GMT+08:00
Description:	--		

Optionally, you can use the following code to list all the images:

```
def list_images(conn):
    print("List Images:")
    for image in conn.image.images():
        print(image)
```

Using Python OpenStack SDK to Create a Private Image

IMS supports the native OpenStack Glance v2 image API, with which you can create a private image with your image file. The supported image types are VHD, ZVHD, QCOW2, and VMDK.

The following uses the image in the QCOW2 format as an example. Image uploading takes a long time, which depends on the image size and network quality.

```
def upload_image(conn):
    #upload the image
    img =conn.image.upload_image(
        name='name',
        disk_format='qcow2',
        container_format='bare',
        properties='{"description": "cirros image"}',
        min_disk=4,
        data=open('cirros.img', 'rb')
    )
    # wait until the image to be active status.
    activeFlag = False
    i = 1
    while(i < 10):
        status =conn.image.get_image(img.id).status
        print status
        if(status == 'active'):
            activeFlag = True
            break;
            i = i + 1
            sleep(60)
        if( not activeFlag):
            print 'Image upload failed'
```

Table 2-15 Parameter description

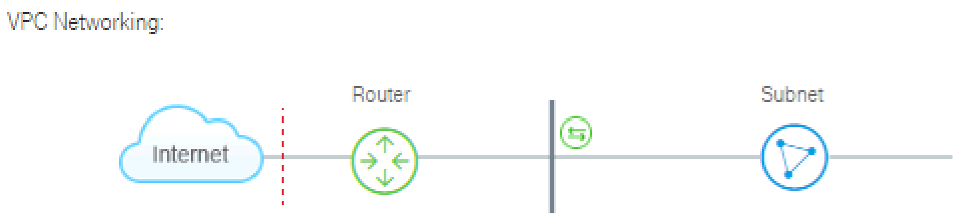
Parameter	Description	Example Value
diskFormat	Specifies the disk format.	qcow2 IMS supports VHD, ZVHD, QCOW2, and VMDK images.
data	Specifies the image file to be uploaded.	open('cirros.img', 'rb') **cirros.img** is the name of the image file to be uploaded.

2.3.6 VPC Python SDK Demo

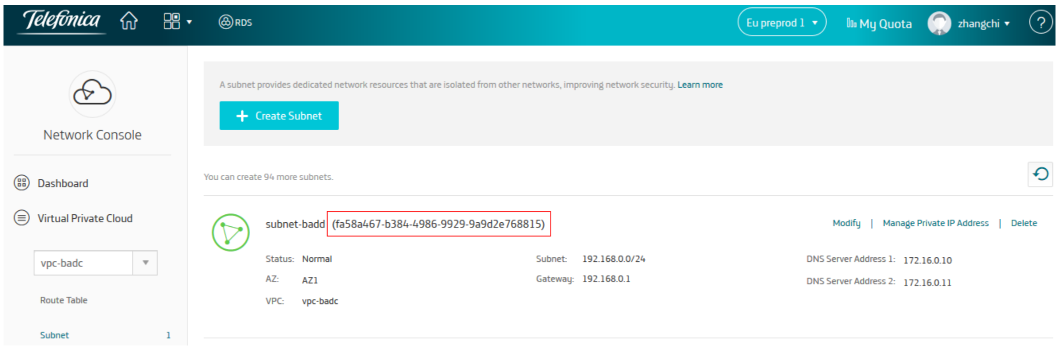
VPC Service Python OpenStack SDK Demo

VPC enables you to provision logically isolated, configurable, and manageable virtual networks for ECSs, improving security of cloud resources and simplifying network deployment.

A typical VPC is composed of a router, network, and subnet, as shown in the following figure.



You can create a VPC on the console and obtain the UUID.



A subnet provides dedicated network resources that are isolated from other networks, improving network security. Learn more

You can create 94 more subnets.

subnet-badd (fa58a467-b384-4986-9929-9a9d2e768815)

Status: Normal
AZ: AZ1
VPC: vpc-badc
Subnet: 192.168.0.0/24
Gateway: 192.168.0.1
DNS Server Address 1: 172.16.0.10
DNS Server Address 2: 172.16.0.11

- Router: A router is a logical entity for forwarding packets across internal subnets and translating the IP addresses of the packets on external networks through an appropriate external gateway.
- Network: A network is an isolated layer-2 network segment, which is similar to a VLAN in the physical network.
- Subnet: A subnet is an IP address segment consisting of IPv4 or IPv6 addresses with their associated configuration status.

Creating a VPC and Subnet

Python OpenStack SDK allows you to create a subnet. The detailed operations are as follows:

1. Create a router.
2. Create a network.
3. Create a subnet.
4. Connect the subnet to the router.

The following code shows the network creation process. You can modify these configurations as required. After you have created the router, network, and subnet and connected the subnet to the router, a new VPC is displayed on the console.

```
def create_VPC(conn) :
   #create a router
   testRouterName = "PythonSDKVPC"
   router = conn.network.create_router(
      name=testRouterName
      )
   #create_network
   testNetworkName = "PythonSDKNet"
   network = conn.network.create_network(
      name=testNetworkName
      )
   #create a subnet
   testSubnetName = "PythonSDKSubnet"
   subnet = conn.network.create_subnet(
      name = testSubnetName,
      is_dhcp_enabled = True,
      cidr = "192.168.1.0/24",
      network_id = network.id
      )
   #connect the subnet to the router, make the subnet connect to the internet.
   conn.network.add_interface_to_router(router, subnet.id)
```

Deleting a VPC

Before deleting a VPC, you need to delete ECSs created in the subnet in the VPC, cancel the association between the router and the subnet, and delete the subnet and the network.

After the ECS deletion command is executed, you can delete the network after ensuring that the ECSs are deleted based on the ECS deletion status. You can use the following code to delete the network:

```
def deleteVPC(conn):
   conn.network.remove_interface_from_router(router, subnetId)
   conn.network.delete_subnet(subnetId)
   conn.network.delete_network(networkdId)
   conn.network.delete_router(routerId)
```

External Network

An external network is a network with attribute **router:external** set to **true**. This network is used to allocate an EIP. After the EIP is bound to an ECS, the ECS can be accessed from the Internet.

An external network is already available and you do not need to create one.

You can use the following code to get the external network:

```
def get_external_network(self):
    for network_each in self.conn.network.networks():
        if network_each.is_router_external == True :
          return network_each
```

2.3.7 ECS Python SDK Demo

Creating an ECS

1. Obtain a flavor ID.

   You can use the following code to query all flavors and use a qualified flavor ID to create an ECS:

   ```
   def list_flavors(conn):
      print("List Flavors:")
      for flavor in conn.compute.flavors():
         print(flavor)
   ```

2. Create a security group.

   For details about how to create a security group, see section 2.4.2 How Can I Create a Security Group?.

   Optionally, you can create a security group using Python OpenStack SDK based on the following code:

   ```
   def create_security_groups(conn):
      #create_SG
      testSGName = "PythonSDKSG"
      createdSG = conn.network.create_security_group(
         name = testSGName
         )
      # open a port.
      conn.network.security_group_open_port(createdSG.id, 8080,    protocol='tcp')
      #allow ping
      conn.network.security_group_allow_ping(createdSG.id)
      # More detailed rules
      IPV4 = 'IPv4'
      PROTO = 'tcp'
      PORT = 22
      DIR = 'ingress'
      conn.network.create_security_group_rule(
         direction=DIR,
         ethertype=IPV4,
         port_range_max=PORT,
         port_range_min=PORT,
         protocol=PROTO,
         security_group_id=createdSG.id
         )
   ```

3. Create a key pair.

   For details about how to create a private key pair, see section 2.4.1 How Can I Create a Key Pair on the Console?.

   Optionally, you can create a key pair using Python OpenStack SDK based on the following code:

```
def create_keypair(conn):
    keypair = conn.compute.find_keypair(KEYPAIR_NAME)
    if not keypair:
        print("Create Key Pair:")
        keypair = conn.compute.create_keypair(name=KEYPAIR_NAME)
        print(keypair)
        try:
            os.mkdir(SSH_DIR)
        except OSError as e:
            if e.errno != errno.EEXIST:
                raise e
        with open(PRIVATE_KEYPAIR_FILE, 'w') as f:
            f.write("%s" % keypair.private_key)
        os.chmod(PRIVATE_KEYPAIR_FILE, 0o400)
    return keypair
```

4. Create an ECS.

   You can use the following code to create an ECS. You can use interface wait_for_serve() to continuously query the ECS status until the ECS is in the specified status or the query times out. You can modify the parameters as required. In the following example, the timeout interval is 4 minutes by default.

```
def create_vm(conn):
    server = conn.compute.create_server(
       name='server_name', flavor_id='flavorId', image_id='imageID',
key_name='keypairName',networks=[{"uuid": 'networkId'}])
    conn.compute.wait_for_server(server)
```

Table 2-16 Parameter description

Parameter	Description	Example Value
flavorId	Specifies the ID of the flavor.	normal2
imageId	Specifies the ID of the image.	51b2c37f-f5bd-40e0-8aa2-1899a6bbca30
keypairName	Specifies the key pair name.	mykeypair

Binding an EIP to an ECS

1. Query the ECS port ID.

```
def get_vm_port_id(conn):
    ifs = list(conn.compute.server_interfaces(self.server))
    port_id = ifs[0].port_id
```

2. Create the EIP.

```
def createEIP(conn):
    eip = conn.network.create_ip(floating_network_id='external_network_id
',port_id='port_id')
    count = 1
    createFlag = False
    while(count < 10):
        if(conn.network.get_ip(fip.id).status == 'ACTIVE'):
            createFlag = True
            print 'eip created success'
            break;
        count = count + 1
        sleep(1)
    if(not createFlag):
         print 'eip create failed'
```

external_network_id indicates the external network ID. For details, see section External Network.

The EIP is automatically bound to the ECS identified by the ECS port ID specified in the preceding code.

Unbinding an EIP from an ECS

You can unbind an EIP from an ECS based on the following code.

The unbinding operation takes several seconds, and the EIP can be deleted only after it is unbound. An error message may be reported if you delete an EIP during the unbinding operation.

```
def disassociate_eip(conn):
    conn.compute.remove_floating_ip_from_server(server,
eip.floating_ip_address)
    disCount = 1
    removeFlag = False
    while(disCount < 10):
        if(conn.network.get_ip(eip.id).status == 'DOWN'):
            removeFlag = True
            print 'eip disassociate success'
            break;
         count = count + 1
         sleep(1)
    if(not removeFlag):
            print 'eip disassociate failed'
```

Deleting an ECS

The ECS deletion process takes several seconds.

```
def delete_server(conn):
    conn.compute.delete_server('serverId')
    conn.compute.wait_for_delete(cls.server)
```

Querying the ECS Status

Use the following code to query the ECS status:

```
def get_ecs(conn):
    print conn.compute.get_server('serverID').status
```

(Optional) Modifying the ECS Flavor

After the flavor is modified, you can roll back the modification or make the modification take effect.

1. Modify the flavor.

```
def resize_server(conn):
    conn.compute.resize_server(server,'resizeFlavorId')
    conn.compute.wait_for_server(server,"VERIFY_RESIZE")
```

2. Confirm the modification.

```
conn.compute.confirm_server_resize(server)
```

3. Roll back the modification. Rollback cannot be performed if you have already made the modification take effect.

```
conn.compute.revert_server_resize(server)
```

Restarting an ECS

Use the following code to restart an ECS:

```
def reboot_server(conn):
    conn.compute.reboot_server(server,'rebootType')
```

The **rebootType** value can be **HARD** or **SOFT**.

Stopping an ECS

Use the following code to stop an ECS:

```
def stop_server(conn):
    conn.compute.stop_server(server)
```

2.3.8 EVS Python SDK Demo

EVS disks are scalable virtual block storage devices designed based on the distributed architecture. You can create EVS disks online and attach them to ECSs. The method for using EVS disks is the same as that for using hard disks on physical servers. Compared with traditional hard disks, EVS disks have higher data reliability and I/O throughput capabilities. They are also easier to use. EVS disks apply to file systems, databases, and system software and applications that require block storage devices.

Creating a Volume

You can create a volume using Python OpenStack SDK based on the following code. The volume can be attached to the ECS only when the volume is in the **available** status.

```
def create_volume(conn):
    volume = conn.block_store.create_volume(
        name='volume_name',
        size=1)
    conn.block_store.wait_for_status(volume,
                                status='available',
                                failures=['error'],
                                 interval=2,
                                 wait=120)
```

Attaching a Volume to an ECS

You can attach a volume to an ECS using Python OpenStack SDK based on the following code. The attachment operation is successful after the volume is in the **in-use** status.

```
def attach_volume_to_ecs(conn):
   attach_attrs = {
      'volume_id': attach_volume_id
      }
   attachment = conn.compute.create_volume_attachment(server,
**attach_attrs)
   conn.block_store.wait_for_status(volume,
                                    status='in-use',
                                    failures=['error'],
                                    interval=2,
                                    wait=120)
```

Detaching a Volume from an ECS

You can detach a volume from an ECS based on the following code. The detachment process takes several seconds, and the volume can be deleted only after the volume is detached.

```
def attach_volume_to_ecs(conn):
   conn.compute.delete_volume_attachment(attachment,server)
   conn.block_store.wait_for_status(volume,
                                    status='available',
                                    failures=['error'],
                                    interval=2,
                                    wait=120)
```

2.3.9 RTS Python SDK Demo

Preparing a Heat Template

A Heat template describes the infrastructure for a cloud application in a text file that is readable and writable by humans.

Prepare the Heat template file, for example, **heatTemplate.yaml**, as shown in the following.

```
#
# Minimal HOT template defining a single compute server.
#
heat_template_version: 2013-05-23

description: >
  Minimal HOT template for stack

parameters:
  key_name:
    type: string
    description: Name of an existing key pair to use for the server
    constraints:
      - custom_constraint: nova.keypair
  flavor:
    type: string
    description: Flavor for the server to be created
    default: s1.large
    constraints:
      - custom_constraint: nova.flavor
  image:
    type: string
    description: Image ID or image name to use for the server
    constraints:
      - custom_constraint: glance.image
  network:
    type: string
    description: Network used by the server

resources:
  server:
    type: OS::Nova::Server
    properties:
      key_name: { get_param: key_name }
      image: { get_param: image }
      flavor: { get_param: flavor }
      networks: [{network: {get_param: network} }]

outputs:
  server_networks:
    description: The networks of the deployed server
    value: { get_attr: [server, networks] }
```

Creating a Stack

The templates enable the creation of most OpenStack resource types, such as instances, EIPs, volumes, security groups, and users. The resources, once created, are referred to as stacks.

You can use the following code to create a stack:

```
def create_heat_stack():
    tname = "hello_world.yaml"
    with open(tname) as f:
        template = f.read()
        parameters = {
            'image': 'imageId',
            'key_name': 'keyname',
            'network': 'networkId',
        }
        sot = cls.conn.orchestration.create_stack(
            name='stackName',
            parameters=parameters,
            template=template,
        )
        conn.orchestration.wait_for_status(
            sot, status='CREATE_COMPLETE', failures=['CREATE_FAILED'])
```

Deleting a Stack

You can delete a stack based on the following code:

```
def delete_stack(conn):
    conn.orchestration.delete_stack(stack)
```

2.3.10 AS Python SDK Demo

Creating an AS Group

An AS group is a set of ECSs with the same application scenario configurations. An AS group defines the minimum and maximum numbers of ECSs.

You can create an AS group based on the following code, where **vpc_id**, **networks**, and **security_groups** are mandatory parameters. Before creating an AS group, you must create a VPC as well as networks and security groups in this VPC.

```
def test_creat_group(self):
_group = {
       "name": "as_NameTest_modify",
       "scaling_configuration_id": "33b55531-78f8-43c9-8cf5-ffec40bd0c6f",
       "desire_instance_number": 10,
       "min_instance_number": 2,
       "max_instance_number": 10,
       "cool_down_time": 200,
       "lb_listener_id": "114863c227a64255bd25157e0beb783c",
       "available_zones": ["eu-de-02"],
       "health_periodic_audit_method": "NOVA_AUDIT",
       "health_periodic_audit_time": "15",
       "instance_terminate_policy": "OLD_CONFIG_NEW_INSTANCE",
       "vpc_id": "2f2b426c-2072-47a7-babc-c35080fa79d4",
       "networks": [{
         "id": "f80308f4-2608-4ae8-9489-c87720383ae5"
       }],
       "notifications": ["EMAIL"],
       "delete_publicip": "true",
       "security_groups": [{
         "id": "57f0a6cd-c427-4e40-a9a2-301ca90893fd"
      }]
     }
     group = self.conn.auto_scaling.create_group(**_group)
     group = self.conn.auto_scaling.get_group(group)
     logging.info(group.id)
```

Creating an AS Configuration

An AS configuration defines the configurations for creating instances in an AS group. The AS service automatically adds instances to an AS group based on the AS configuration.

You can create an AS configuration based on the following code. When using an existing ECS flavor as the template to create the AS configuration, specify parameter **instance_id**. In this case, parameter **flavorRef**, **imageRef**, and **disk** do not take effect. If **instance_id** is not specified, **flavorRef**, **imageRef**, and **disk** are mandatory.

```
def test_creat_config(self):
   instance_config = {
     "flavor_id": "normal1",
     "image_id": "ba391176-5e4c-4c06-8466-349f6b5fc91b",
     "disk": [{
        "size": 40,
        "volume_type": "SATA",
        "disk_type": "SYS"
```

```
         }],
        "metadata": {
            "key1": "value1",
            "tag": "app"
        },
        "key_name": "KeyPair-0406-as",
        "user_data": "wewfef46565",
        "public_ip": {
            "eip": {
                "ip_type": "5_bgp",
                "bandwidth": {
                    "size": 10,
                    "share_type": "PER",
                    "charging_mode": "traffic"
                }
            }
        }
        }
        config_name = "auto-scaling-config-name"
        config = self.conn.auto_scaling.create_config(config_name, **instance_config)
        config = self.conn.auto_scaling.get_config(config)
        logging.info(config.id)
```

Creating an AS Policy

The AS service supports the periodic, scheduled, and alarm policies. The periodic policy can be configured as daily, weekly, or monthly. If you configure the alarm policy, the selected or created alarm policies can be associated with only one AS group.

You can create a scheduled policy (daily) using OpenStack4j based on the following code:

```
def test_creat_policy_Daily(self):
    as_group_id = "196ddd9c-e1f2-4088-b150-b67ae5ebf746"
    as_policy_name = "as-policy-name"
    _policy = {
         "name": as_policy_name,
         "scaling_policy_action": {
             "operation": "ADD",
             "instance_number": 1
         },
         "cool_down_time": 900,
         "scheduled_policy": {
             "launch_time": "16:00",
             "recurrence_type": "Daily",
             "recurrence_value": None,
             "start_time": "2017-07-14T03:34Z",
             "end_time": "2017-07-27T03:34Z"
         },
         "type": "RECURRENCE",
         "scaling_group_id": as_group_id
    }
    policy = self.conn.auto_scaling.create_policy(**_policy)
    policy = self.conn.auto_scaling.get_policy(policy)
    logging.info(policy)
```

Creating a Lifecycle Hook

The purpose of adding a lifecycle hook to the AS group is to suspend the instance status to **Wait (Adding to AS group)** or **Wait (Removing from AS group)** during a scaling action. This status retains until the suspension times out or you manually call back the action.

Code reference:

```
 #test_create lifecycle_hook
 def test_create_life_cycle_hook(self):
    groupID = "58cbfcab-ebc6-4263-8b71-7d414810488d"
    groupID1 = "936634ad-1a4b-4929-b574-2bc2cbacb608"
    attrs = {
       "lifecycle_hook_name": "test-hook_c",
       "lifecycle_hook_type": "INSTANCE_TERMINATING",
       "default_result": "ABANDON",
       "default_timeout": "",
       "notification_topic_urn":
"urn:smn:cn-suzhou2-1:ebac0c927c104c4587687ce375d0b656:as_test",
       "notification_metadata": "xxxxxxxx"
    }
    hook = self.conn.auto_scaling.create_lifecycle_hook(groupID,**attrs)
```

2.3.11 Cloud Eye Python SDK Demo

Querying Metrics

You can query the metric list in the system and specify the namespace, metric name, dimension, sorting order, start records, and the maximum number of records to filter the search result.

Use the following code to obtain all metrics of the current tenant:

```
query = {
"namespace": "SYS.ECS",
"metric_name": "cpu_util",
"dimensions": [{
"name": "instance_id",
"value": "d9112af5-6913-4f3b-bd0a-3f96711e004d"
}],
"order": "desc",
"marker": "SYS.ECS.cpu_util.instance_id:9f31d05a-76d5-478a-b864-b1b5e8708482",
"limit": 10
}
# get some metric
metrics = conn.cloud_eye.metrics(**query)
```

Parameter	Description	Example Value
namespace	Specifies the namespace, for example, the ECS namespace.	SYS.ECS
metric_name	Specifies the metric name.	disk_read_bytes_rate
dim	Specifies the metric dimension. A maximum of three dimensions are supported, and the dimensions are	AutoScalingGroup,ca3fb7aa-da18-4abc-8206-630cbbb74e14

Parameter	Description	Example Value
	numbered from 0 in the **dim.{i}=key,value** format.	
start	Specifies the paging start value. The format is **namespace.metric_name.key:value**.	SYS.ECS.cpu_util.instance_id:d9112af5-6913-4f3b-bd0a-3f96711e004d
limit	The value ranges from **0** to **1000** (**0** excluded and **1000** included). The default value is **1000**. This parameter is used to limit the number of search results.	50
order	Specifies the sorting order of search results. The value can be **asc** (ascending order) or **desc** (descending order). The default value is **desc**.	desc

Querying the Alarm Rule List

You can query alarm rules and specify the paging parameters to limit the number of search results displayed on a page. You can also set the sorting order of search results.

```
query = {
"limit": 1,
"marker": "last-alarm-id",
"order": "desc"
}
# get some alarm
for alarm in conn.cloud_eye.alarms(**query):
logging.info(alarm)
```

Parameter	Description	Example Value
start	Specifies the first queried alarm to be displayed on a page. The value is **alarm_id**.	al1498535073312Z27eznaxV
limit	The value ranges from **0** to **100** (**0** excluded and **100** included). The default value is **100**. This parameter is used to limit the number of search results.	50
order	Specifies the sorting order of search results. The value can be **asc** (ascending order) or **desc** (descending order). The default value is **desc**.	desc

Querying an Alarm Rule

You can query the alarm rule based on the alarm ID.

```
# plain ID
alarm = conn.cloud_eye.get_alarm("some-alarm-id")
# Instance with ID
alarm = conn.cloud_eye.get_alarm(alarm.Alarm(id="some-alarm-id"))
```

Parameter	Description	Example Value
alarm_id	Specifies the alarm rule ID.	al1498535073312Z27eznaxV

Enabling or Disabling an Alarm Rule

You can enable or disable an alarm rule.

```
#start alarm
conn.cloud_eye.enable_alarm("some-alarm-id")
or
conn.cloud_eye.enable_alarm(alarm.Alarm(id="some-alarm-id"))
# stop alarm
conn.cloud_eye.disable_alarm("some-alarm-id")
or
conn.cloud_eye.disable_alarm(alarm.Alarm(id="some-alarm-id"))
```

Parameter	Description	Example Value
alarm_id	Specifies the alarm rule ID.	al1498535073312Z27eznaxV

Deleting an Alarm Rule

You can delete an alarm rule.

```
conn.cloud_eye.delete_alarm("some-alarm-id")
or
conn.cloud_eye.delete_alarm(alarm.Alarm(id="some-alarm-id"))
```

Parameter	Description	Example Value
alarm_id	Specifies the alarm rule ID.	al1498535073312Z27eznaxV

Querying Monitoring Data

You can query the monitoring data at a specified granularity for a specified metric in a specified period of time. You can specify the dimension of data to be queried.

```
def get_epoch_time(datetime_):
if datetime_:
seconds = time.mktime(datetime_.timetuple())
return int(seconds) * 1000
else:
return None
now = datetime.datetime.now()
_to = now
_from = now - datetime.timedelta(minutes=5)
```

```
query = {
"namespace": "MINE.APP",
"metric_name": "cpu_util",
"from": get_epoch_time(_from),
"to": get_epoch_time(_to),
"period": 300,
"filter": "average",
"dimensions": [{
"name": "instance_id",
"value": "33328f02-3814-422e-b688-bfdba93d4050"
}]
}
for aggregation in conn.cloud_eye.metric_aggregations(**query):
logging.info(aggregation)
```

Parameter	Description	Example Value
namespace	Specifies the namespace, for example, the ECS namespace.	SYS.ECS
metric_name	Specifies the metric name.	disk_read_bytes_rate
from	Specifies the start time of the query. The value is a UNIX timestamp and the unit is ms. Set the value of **from** to at least one period earlier than the current time. Rollup aggregates the raw data generated within a period to the start time of the period. Therefore, if values of **from** and **to** are within a period, the query result will be empty due to the rollup failure. You are advised to set **from** to be at least one period earlier than the current time. Take the 5-minute period as an example. If it is 10:35 now, the raw data generated between 10:30 and 10:35 will be aggregated to 10:30. Therefore, in this example, if the value of **period** is 5 minutes, the value of **from** should be 10:30 or earlier. **NOTE** Cloud Eye rounds up the value of **from** based on the granularity required to perform the rollup.	14991341910611
to	Specifies the end time of the query. The value is a UNIX timestamp and the unit is ms. The value of parameter **from** must be earlier than that of parameter **to**.	14991341892581
period	Specifies the data monitoring granularity.	Value range: • **1**: The data is monitored in real time. • **300**: The data monitoring granularity is 5 minutes. • **1200**: The data monitoring

Parameter	Description	Example Value
		granularity is 20 minutes. • **3600**: The data monitoring granularity is 1 hour. • **14400**: The data monitoring granularity is 4 hours. • **86400**: The data monitoring granularity is 1 day.
filter	Specifies the data rollup method.	max, min, average, sum, variance
dim	Specifies the metric dimension. A maximum of three dimensions are supported, and the dimensions are numbered from 0 in the **dim.{i}=key,value** format.	AutoScalingGroup,ca3fb7aa-da18-4abc-8206-630cbbb74e14

Adding Monitoring Data

You can add one or multiple pieces of monitoring data.

```
def get_epoch_time(datetime_):
if datetime_:
seconds = time.mktime(datetime_.timetuple())
return int(seconds) * 1000
else:
return None

now = datetime.datetime.now()
collect_time_1 = now
collect_time_2 = now - datetime.timedelta(minutes=5)
data = [
{
"metric": {
"namespace": "MINE.APP",
"dimensions": [
{
"name": "instance_id",
"value": "33328f02-3814-422e-b688-bfdba93d4050"
}
],
"metric_name": "cpu_util"
},
"ttl": 604800,
"collect_time": get_epoch_time(collect_time_1),
"value": 60,
"unit": "%"
},
{
```

```
"metric": {
"namespace": "MINE.APP",
"dimensions": [
{
"name": "instance_id",
"value": "33328f02-3814-422e-b688-bfdba93d4050"
}
],
"metric_name": "cpu_util"
},
"ttl": 604800,
"collect_time": get_epoch_time(collect_time_2),
"value": 70,
"unit": "%"
}
]
conn.cloud_eye.add_metric_data(data)
```

Parameter	Description	Example Value
metric	Specifies the metric data.	Value in the JSON structure
namespace	Specifies the namespace in the **service.item** format. **service** and **item** each must be a string of 3 to 32 characters starting with a letter and consisting of uppercase letters, lowercase letters, digits, and underscores (_). In addition, **service** cannot be set to **SYS**.	ABC.ECS
metric_name	Specifies the metric name, which must be a string of 1 to 64 characters starting with a letter and consisting of uppercase letters, lowercase letters, digits, and underscores (_).	disk_read_bytes_rate
dimensions	Specifies the list of metric dimensions. Each dimension is a JSON object, and its structure is as follows: **dimension.name**: specifies the dimension name. The value must be a string of 1 to 32 characters starting with a letter and consisting of uppercase letters, lowercase letters, digits, underscores (_), and hyphens (-). **dimension.value**: specifies the dimension value. The value must be a string of 1 to 64 characters starting with a letter and consisting of uppercase letters, lowercase letters, digits, underscores (_), and hyphens (-).	instance_id:33328f02-3814-422e-b688-bfdba93d4050
ttl	Specifies the data validity period. The unit is second. The maximum value is 604,800 seconds. If the validity period expires, the data will be automatically deleted.	172800
collect_time	Specifies the time when the data was collected. The time is UNIX timestamp (ms) format.	1502938466458

Parameter	Description	Example Value
	NOTE Since there is a latency between the client and the server, the data timestamp to be inserted should be within the period that starts from three days before the current time plus 20s to 10 minutes after the current time minus 20s. In this way, the timestamp will be inserted to the database without being affected by the latency.	
value	Specifies the metric value.	60
unit	Specifies the data unit.	B
type	Specifies the data type. The value can only be **int** or **float**.	**int** or **float**

Querying Quotas

You can query the total number of resource quotas that can be created and the quota usage. Currently, the resource type can be only the alarm rule.

```
quotas = conn.cloud_eye.quotas()
for quota in quotas:
logging.info(quota)
```

2.3.12 DNS Python SDK Demo

Service Description

Domain Name Service (DNS) provides highly available and scalable authoritative DNS resolution services and domain name management services. It translates domain names or application resources into IP addresses required for network connection. By doing so, visitors' access requests are directed to the desired resources.

Creating a Private Zone

To use the DNS service to manage domain names in VPCs, you need to configure private zones on the DNS console. You can use the Python OpenStack SDK to create a private zone as follows:

1. Specify the VPC to be associated.
2. Create a private zone.

You can create a private zone using the Python OpenStack SDK based on the following code. After the private zone is created, it will be displayed on the private zone page of the DNS console.

```
def setUpClass(cls):
    super(TestZone, cls).setUpClass()
    # get a router
    routers = cls.conn.network.routers(limit=2)
    idx = 0
    for _router in routers:
        idx += 1
        print _router
        if idx == 1:
            cls.router = _router
        if idx == 2:
            cls.router2 = _router
            break
    # create zone
    cls.zone = auto_create_private_zone(cls.conn, cls.NAME, cls.router.id,region)
```

Associating a VPC

You can use the Python OpenStack SDK to associate a private zone with a VPC on the cloud platform. The association procedure is as follows:

1. Specify the VPC to be associated.
2. Select the target private zone to associate with the VPC.

You can associate a private zone with a VPC using the Python OpenStack SDK based on the following code:

```
def add_router_to_zone(self):
    # Designate a router
    resource2.wait_for_status(self.conn.dns._session, self.zone, "ACTIVE", interval=5,
failures=["ERROR"])
    # Associate the private zone to the router
    result = self.conn.dns.add_router_to_zone(self.zone, **{"router_id":
self.router2.id,"router_region": region})
    self.assertEqual(result.router_id, self.router2.id)
    self.assertEqual(result.router_region, region)
    zone = self.conn.dns.get_zone(self.zone)
    self.assertEqual(2, len(zone.routers))
    router_ids = [_router["router_id"] for _router in zone.routers]
self.assertIn(self.router.id, router_ids)
```

Disassociating a VPC

You can use the Python OpenStack SDK to disassociate a private zone from a VPC on the cloud platform. The code is as follows:

```
def remove_router_of_zone(self):
    resource2.wait_for_status(self.conn.dns._session, self.zone, "ACTIVE", interval=5,
failures=["ERROR"])
    result = self.conn.dns.remove_router_from_zone(self.zone, **{        "router_id":
self.router.id,
      "router_region": region
    })
    self.assertEqual(result.router_id, self.router.id)
    self.assertEqual(result.router_region, region)
```

Deleting a Private Zone

You can delete a private zone that you do not need to manage using the DNS service. After the deletion, domain names included in this zone cannot be resolved.

Before deleting a private zone, ensure that all record sets in this zone have been backed up. The code is as follows:

```
def tearDownClass(cls):
   # delete zone
   cls.conn.dns.delete_zone(cls.zone)
```

2.3.13 ELB Python SDK Demo

Creating a Load Balancer

You can create a load balancer using the Python OpenStack SDK based on the following code:

```
def create_load_balancer(self, **attrs):
   """Create a new load balancer from attributes

   :param dict attrs: Keyword arguments which will be used to create
a :class:`~openstack.load_balancer.v1.load_balancer.LoadBalancer`,
      comprised of the properties on the LoadBalancer class.         :returns: a
asynchronous LoadBalancer job
   :rtype: :class:`~openstack.load_balancer.v1.load_balancer.
LoadBalancerJob`
   """
   return self._create(_lb.LoadBalancerJob, prepend_key=False, **attrs)
```

Creating a Listener

You can create a listener using the Python OpenStack SDK based on the following code. A listener can be created only when a load balancer is available.

```
def create_listener(self, **attrs):
   """Create a new listener from attributes

   :param dict attrs: Keyword arguments which will be used to create
      a :class:`~openstack.load_balancer.v1.listener.Listener`,
      comprised of the properties on the Listener class.

   :returns: a listener instance
   :rtype: :class:`~openstack.load_balancer.v1.listener.Listener`
   """
   return self._create(_listener.Listener, prepend_key=False, **attrs)
```

Performing a Health Check

You can perform a health check using the Python OpenStack SDK based on the following code. The health check can be performed only when a listener is available.

```
def create_health_check(self, **attrs):
   """Create a new health check from attributes

   :param dict attrs: Keyword arguments which will be used to create
      a :class:`~openstack.load_balancer.v1.health_check.HealthCheck`,
      comprised of the properties on the HealthCheck class.

   :returns: A health check instance
```

```
    :rtype: `:class: ~openstack.load_balancer.v1.health_check.HealthCheck`
    """
    return self._create(_hc.HealthCheck, prepend_key=False, **attrs)
```

Adding Members

You can add members to a listener using the Python OpenStack SDK based on the following code:

```
def add_members_to_listener(self, listener, members):
    """Add backend members for a listener

    :param listener: Either the ID of a listener or an instance of
           :class:`~openstack.load_balancer.v1.listener.Listener`
    :param members: list of dicts which contain the server_id and address.
        server_id is ECS service id, address is ECS server internal IP.
        [{"server_id": "dbecb618-2259-405f-ab17-9b68c4f541b0",
          "address": "172.16.0.31"}] for example.

    :return: a operate member job
    :rtype: :class:`~openstack.load_balancer.v1.listener.OperateMemberJob`
    """
    listener = self._get_resource(_listener.Listener, listener)
    return listener.add_members(self._session, members)
```

Creating a Certificate

You can create a certificate using the Python OpenStack SDK based on the following code:

```
def create_certificate(self, **attrs):
    """Create a new certificate from attributes

    :param dict attrs: Keyword arguments which will be used to create
a :class:`~openstack.certificate.v1.certificate.Certificate`,       comprised of the
properties on the Certificate class.
    :returns: a certificate instance
    :rtype: :class:`~openstack.certificate.v1.certificate.Certificate`   """
    return self._create(_cert.Certificate, prepend_key=False, **attrs)
```

2.3.14 VBS Python SDK Demo

Creating a VBS Backup

You can create a VBS backup using the Python OpenStack SDK based on the following code. After the VBS backup is created, it will be displayed in the VBS list on the VBS console.

```
def create_backup(self):
    backup = {
        "volume_id": self.volume.id,
        "name": "sds",
        "description": "created by openstacksdk"
    }

    result = self.conn.volume_backup.create_backup(**backup)
    # assert result.job_id != None
    self.job_id = result.id
```

Request parameter description:

Parameter	Mandatory	Type	Description
backup	Yes	dict	Specifies the backup to be created.
volume_id	Yes	string	Specifies the ID of the disk to be backed up.
snapshot_id	No	string	Specifies the snapshot ID of the disk to be backed up.
name	Yes	string	Specifies the backup name. The value is a string of 1 to 64 characters consisting of digits, letters, underscores (_), and hyphens (-).
description	No	string	Provides supplementary information about the backup. The value is a string of 1 to 64 characters and cannot contain the less-than sign (<) or greater-than sign (>).

Querying VBS Backup Details

You can query the backup list and obtain the backup details using the Python OpenStack SDK based on the following code:

```
def query_backups_detail(self):
    backups = self.conn.volume_backup.backups(details=True)

    query = {
        "name": "volume-backup-" + self.volume.id,
        # "status": "available",
        "volume_id": self.volume.id,
        # "marker": "some-backup-id",
        "limit": 10
    }
    backups = self.conn.volume_backup.backups(details=True, **query)
    for backup in backups:
        print backup.name
```

Request parameter description:

Parameter	Mandatory	Type	Description
name	No	string	Specifies the name of the backup to be queried. This parameter is used to query the backups whose names are specified character strings.
status	No	string	Specifies the status of the backup to be queried. This parameter is used to query the backups in a specified state. The value can be **available**, **error**, **restoring**, **creating**,

Parameter	Mandatory	Type	Description
			deleting, or **error_deleting**.
offset	No	int	Specifies the offset of the queried details.
limit	No	int	Specifies the maximum number of query results that can be returned.
volume_id	No	string	Specifies the disk ID of the backup to be queried. This parameter is used to query the backups for specific disks.

Restoring a Disk Using a VBS Backup

You can restore a disk from a VBS backup using the Python OpenStack SDK based on the following code:

```
def restore_backup(self):
    self.query_backups()
    return self.conn.volume_backup.restore_backup(self.backup_id, self.volume.id)
```

Request parameter description:

Parameter	Mandatory	Type	Description
restore	Yes	dict	Specifies the operation of restoring the disk using a backup.
backup_id	Yes	string	Specifies the ID of the backup used to restore a disk.
volume_id	Yes	string	Specifies the ID of the disk to be restored.

Deleting a Backup

You can delete a backup using the Python OpenStack SDK based on the following code:

```
def delete_backup(self):
    self.query_backups()
    self.conn.volume_backup.delete_backup(self.backup_id)
```

Request parameter description:

Parameter	Mandatory	Type	Description
tenant_id	Yes	string	Specifies the ID of the tenant.
backup_id	Yes	string	Specifies the ID of the backup used to restore a disk.

Creating a Backup Policy

You can create a backup policy using the Python OpenStack SDK based on the following code:

```
def create_policy(self):
   data = {
      "remain_first_backup_of_curMonth": True,
      "rentention_num": 10,
      "frequency": 1,
      "start_time": "12:00",
      "status": "ON"
   }
   volume_backup_name = "SDK-backup-test-1"
   policy = self.conn.volume_backup.create_backup_policy(volume_backup_name, **data)
   print policy
```

Request parameter description:

Parameter	Mandatory	Type	Description
backup_policy_name	Yes	string	Specifies the backup policy name. The name is a string of 1 to 64 characters consisting of letters, digits, underscores (_), and hyphens (-). It cannot start with **default**.
scheduled_policy	Yes	dict	Specifies details about the scheduling policy.
start_time	Yes	string	Specifies the backup start time, which needs to be converted into the local UTC time (on the hour only). The value is in **HH:mm** format.
frequency	No	integer	Specifies the backup interval (1 to 14 days). Select either this parameter or **week_frequency**. If you select both, this parameter is used.
week_frequency	No	list<dict>	Specifies on which days of each week backup jobs are executed. The value can be one or more of the following: SUN, MON, TUE, WED, THU, FRI, SAT
rentention_num	No	integer	Specifies the retained number (minimum: 2) of backups. Select either this parameter or **rentention_day**. If you select both, this parameter is used.
rentention_day	No	integer	Specifies how many days backups are retained.

Parameter	Mandatory	Type	Description
remain_first_backup_of_curMonth	Yes	string	Specifies whether to retain the first backup in the current month. The value can be **Y** or **N**.
status	Yes	string	Specifies the backup policy status. The value can be **ON** or **OFF**.

Deleting a Backup Policy

You can delete a backup policy using the Python OpenStack SDK based on the following code:

```
def delete_policy(self):
   policy_id = self.query_policies().id
   self.conn.volume_backup.delete_backup_policy(policy_id)
```

Request parameter description:

Parameter	Mandatory	Type	Description
tenant_id	Yes	string	Specifies the ID of the tenant.
policy_id	Yes	string	Specifies the ID of the policy.

Querying Backup Policies

You can query backup policies using the Python OpenStack SDK based on the following code:

```
def query_policies(self):
   policies = list(self.conn.volume_backup.backup_policies())
   if policies and len(policies) > 0:
      return policies[0]
```

2.3.15 CTS Python SDK Demo

Tracker

A tracker will be created after CTS is enabled. All traces recorded by CTS are associated with the tracker.

Creating a Tracker

You can use the Python OpenStack SDK to create a tracker on the cloud platform.

You can create a tracker using the Python OpenStack SDK based on the following code by specifying required parameters, such as the bucket name, folder name, and status:

```
tracker=conn.cts.create_tracker(bucket_name="obs-ting",file_prefix_name='SDKunitte
st');
```

Deleting a Tracker

You can delete a tracker using the Python OpenStack SDK based on the following code:

```
conn.cts.delete_tracker(tracker='system', ignore_missing=True);
```

Updating a Tracker

You can update a tracker using the Python OpenStack SDK based on the following code. The information that can be updated includes the bucket name, folder name, and status.

```
tracker=conn.cts.update_tracker(tracker='system',bucket_name="obs--bce0",file_pref
ix_name='SDKunittest' );
```

Querying a Tracker

You can query a tracker using the Python OpenStack SDK based on the following code by specifying the tracker name:

```
tracker=conn.cts.get_tracker(name='system');
```

Trace

This interface is used to query records of operations on resources during the last seven days.

Querying the Trace List

You can query a trace list using the Python OpenStack SDK based on the following code. You can filter out required traces by specifying multiple parameters.

```
list=conn.cts.traces(tracker='system',service_type='CTS',limit='5');
```

2.3.16 SMN Python SDK Demo

Creating a Topic

You can create a maximum of 3000 topics. APIs are idempotent. If a topic of the same name already exists, the status code 200 is returned. Otherwise, the status code 201 is returned.

You can create a topic using the Python OpenStack SDK based on the following code:

```
def operate_topic(conn):
topic_dict = {
'name': 'labj',
'display_name': 'djb',
}
tp = conn.smn.create_topic(**topic_dict)
```

Adding a Subscription

Add a subscription to a specified topic. If the status of the subscription is unconfirmed, a confirmation message is sent to the subscriber. After confirming the subscription, the subscriber can receive notification messages published to the topic. APIs are idempotent. If the added subscription already exists, the status code 200 is returned. Otherwise, the status code 201 is returned.

You can add a subscription using the Python OpenStack SDK based on the following code by specifying **tp** to **topicUrn**:

```
sub_dict = {
'protocol': 'email',
'endpoint': 'xxx@xxx.com',
'remark': 'test',
}
sub = conn.smn.subscript_topic(tp, **sub_dict)
```

Publishing a Message

You can publish messages to a topic. After the message ID is returned, the message has been saved and is to be pushed to the subscribers of the topic. The message format varies depending on the protocol of a subscription.

You can publish a message using the Python OpenStack SDK based on the following code:

```
msg_dict = {
'message': "hello world!"
}
print("publish message")
conn.smn.publish_topic(tp, **msg_dict)
```

2.3.17 MaaS Python SDK Demo

Object Storage Migration Service (MaaSOBS) online migrates OBS of another cloud service provider to the destination cloud platform. This migration service offers secure object authentication, encrypted data transmission, and reliable interruption recovery.

Obtaining the Service Version

You can obtain the current MaaS service version using the Python OpenStack SDK based on the following code:

```
def get_version(conn):
    fip_dversion=conn.maas.versions()
```

Creating a Migration Task

You can create a JSON character string consisting of the migration parameters and then use the JSON string to create a migration task using the Python OpenStack SDK based on the following code. After the migration task is created, it is added to the task queue and waits for execution.

```
def create_task(conn):
    task_dict = {
        "src_node":
        {
           "region": "us-east-1",
           "ak": "AKIAIwww72JCUBV6Addd",
           "sk": " SKIAIwww72JCUBV6Addd123fuxcnk5",
           "object_key":
           {
              "path": "folder-2/",
              "keys": ["transFiles.file"]
           },
           "bucket": "maas-bucket"
        },
        "thread_num": 5,
        "enableKMS": False,
        "dst_node":
        {
           "region": "dst-region01",
           "ak": "asdusac13UFDHASDK1",
           "sk": "yHid7HDJKajdjsyf87658Ih7DFuHDI",
           "object_key": "destination/",
           "bucket": "dst-bucket"
        },
    }
    g_task = conn.maas.create_task(**task_dict)
```

Table 2-17 Parameter description

Parameter	Mandatory	Type	Description	Example Value
region	Yes	String	Specifies the region ID. Note that this parameter is not the region name.	"region": "us-east-1"
ak	Yes	String	Specifies the access key ID. In most cases, the value is a string of 20 characters consisting of digits and letters.	"ak":"AKTAI72RCUBV4AHQO46C"
sk	Yes	String	Specifies the secret access key ID. In most cases, the value is a string of 40 characters consisting of digits and letters.	"sk":"32o6vpFgj76zII1HOad7SrbygHChx9TbwWpDzsHo"
object_key	Yes	JsonString	Specifies the JSON character string of the migration object. In most cases, the value is in the path+keys array. **path** indicates the parent path of the object on the source end. **keys** indicates the object	"object_key": { path: "test-01/", keys: ["10000-files/","rmb001"] }

Parameter	Mandatory	Type	Description	Example Value
			array in the path.	
bucket	Yes	String	Specifies the bucket name. The value is the name of a bucket created by the provider of the public cloud supporting service migration.	"bucket": "maas-bucket"
src_node	Yes	JsonString	Specifies the JSON character string for encapsulating the source-end information.	"src_node":{}
dst_node	Yes	JsonString	Specifies the JSON character string for encapsulating the destination-end information.	"dst_node":{}
thread_num	No	Integer	Specifies the number of threads used for migration. The default value is **5**.	"thread_num":3
description	No	String	Provides supplementary information about the migration task.	"description":"test"
enableKMS	Yes	Boolean	Specifies whether to enable the KMS function to encrypt the object data after the migration.	"enableKMS":True

Pausing a Migration Task

You can pause a migration task using the Python OpenStack SDK based on the following code by specifying the task ID:

```
def stop_task(conn,taskid):
   conn.maas.stop_task(taskid)
```

Table 2-18 Parameter description

Parameter	Mandatory	Type	Description	Example Value
taskid	Yes	String	Specifies the ID of the migration task to be paused. The value is a string of digits.	12345125163

Continuing a Migration Task

You can continue a paused or failed migration task using the Python OpenStack SDK based on the following code by specifying required parameters, including the task ID and the respective AKs and SKs on the source end and destination end:

```
def start_task(conn,taskid):
    srcak="F8ISHCFJ28DK5KV"
    srcsk="AFASFFQTAS2342566SSDfsfd"
    dstak="SDASVNV8ASYSCJASLF"
    dstsk="Qn6FBZFG1Qf6lo17DASDS234SDVJHAFASFFQ"
    conn.maas.start_task(taskid,srcak,srcsk,dstak,dstsk)
```

Table 2-19 Parameter description

Parameter	Mandatory	Type	Description	Example Value
taskid	Yes	String	Specifies the ID of the migration task to be continued. The value is a string of digits.	12345125163
srcak	Yes	String	Specifies the access key ID on the source end for the migration task. In most cases, the value is a string of 20 characters consisting of digits and letters.	AKTAI72RCUBV4AHQO46C
srcsk	Yes	String	Specifies the secret access key ID on the source end for the migration task. In most cases, the value is a string of 40 characters consisting of digits and letters.	32o6vpFgj76zII1HOad7SrbygHChx9TbwWpDzsHo
dstak	Yes	String	Specifies the access key ID on the destination end for the migration task. In most cases, the value is a string of 20 characters consisting of digits and letters.	KEIIFOFUCVDQJ0E0NW4S
dstsk	Yes	String	Specifies the secret access key ID on the destination end for the migration task. In most cases, the value is a string of 40 characters consisting of digits and letters.	32o6vpOg376zII1HOad7SrbyfHChx9TbwwprzsHN

Deleting a Migration Task

You can delete a migration task that is in the wait state using the Python OpenStack SDK based on the following code by specifying the task ID. When **isExpr** is set to **False**, the **openstack.exceptions.ResourceNotFound** exception will be reported if the task does not exist. On the contrary, the exception will not be reported when **isExpr** is set to **True**. The default value is **False**.

```
def delete_task(conn,taskid):
    conn.maas.delete_task(taskid,isExpr)
```

Table 2-20 Parameter description

Parameter	Mandatory	Type	Description	Example Value
taskid	Yes	String	Specifies the ID of the migration task to be deleted. The value is a string of digits.	12345125163
isExpr	Yes	Boolean	**False**: If the task does not exist, the **openstack.exceptions.ResourceNotFound** exception will be reported. **True**: The **openstack.exceptions.ResourceNotFound** exception will not be reported.	False

Querying Details of a Migration Task

You can query details of a migration task using the Python OpenStack SDK based on the following code by specifying the task ID:

```
def get_task_by_id(conn,taskid):
    task=conn.maas.get_task(taskid)
```

Table 2-21 Parameter description

Parameter	Mandatory	Type	Description	Example Value
taskid	Yes	String	Specifies the ID of the migration task to be queried. The value is a string of digits.	12345125163

Querying All Tasks of a Tenant

You can query details of a specified number of migration tasks using the Python OpenStack SDK based on the following code by specifying a JSON character string that consists of three parameters. The first parameter specifies the start number of the migration tasks to be queried, the second parameter specifies the total number of migration tasks to be queried, and the third parameter specifies the status of the migration tasks to be queried. If the task status is not specified, migration tasks of all states will be queried. The following provides an example of querying details 10 latest tasks in the waiting state:

```
def query_tasks(conn):
   query = {
   'start': '0',
   'limit': '10',
   'state':'3'}
   task[] tasklist = conn.maas.tasks(**query):
```

Table 2-22 Parameter description

Parameter	Mandatory	Type	Description	Example Value
start	Yes	Integer	Specifies the start serial number.	"start":1
limit	Yes	Integer	Specifies the maximum number of returned tasks, which cannot exceed 100. Otherwise, the query fails.	"limit":5
state	No	Integer	Specifies the status of the tasks to be queried. If this parameter is left blank, the tasks of all states will be queried. The value can be: **0**: indicates initialized tasks. **1** indicates waiting tasks. **2** indicates executed tasks. **3** indicates paused tasks. **4** indicates failed tasks. **5** indicates successful tasks.	"state":'1' "state":'2' "state":'3' "state":'4' "state":'5'

Querying the Total Number of Migration Tasks

You can query either the total number of migration tasks in a specified state or the total number of migration tasks in all states using the Python OpenStack SDK based on the following code.

Querying the total number of migration tasks in a specified state:

```
def task_count_by_state(conn,state):
   maas_count=conn.maas.task_count(state)
```

Querying the total number of migration tasks in all states:

```
def task_count(conn):
   all_count=conn.maas.task_count()
```

Table 2-23 Parameter description

Parameter	Mandatory	Type	Description	Example Value
state	Yes	Integer	Specifies the status of the tasks to be queried. If this parameter is left blank, the tasks of all states will be queried. The value can be: **0**: indicates initialized tasks. **1** indicates waiting tasks. **2** indicates executed tasks. **3** indicates paused tasks. **4** indicates failed tasks. **5** indicates successful tasks.	"state":'1' "state":'2' "state":'3' "state":'4' "state":'5'

2.3.18 DMS Python SDK Demo

Distributed Message Service (DMS) is a message middleware service based on distributed, high-availability clustering technology. It provides reliable, scalable, fully managed queues for storing messages. DMS enables cloud applications to decouple from each other, achieving high cost-effectiveness.

Creating a Queue

You can create a queue using the Python OpenStack SDK based on the following code. After the queue is created, messages will be sent to this queue.

```
queue_dict = {
             'name': "dmsTestQueue" + self.timeStamp,
             'description': "dmsTestQueue" + self.timeStamp
}
q = conn.dms.create_queue(**queue_dict)
```

Creating a Consumer Group

You can create a consumer group using the Python OpenStack SDK based on the following code. After the consumer group is created, it can consume messages in the queue.

```
groupDict = {
          "groups": [
              {
                  "name": "dmsConsumeGroup" + self.timeStamp
              }
          ]
 }
group = conn.dms.create_groups(queue, **groupDict)
```

Producing Messages

You can produce messages using the Python OpenStack SDK based on the following code:

```
msgDict = {
            "messages": [
                {
                    "body": "testMsg" + self.timeStamp,
                    "attributes":
                    {
                        "attribute1": "value1",
                        "attribute2": "value2"
                    }
                }
            ]
 }
conn.dms.send_messages(queue, **msgDict))
```

Consume Messages

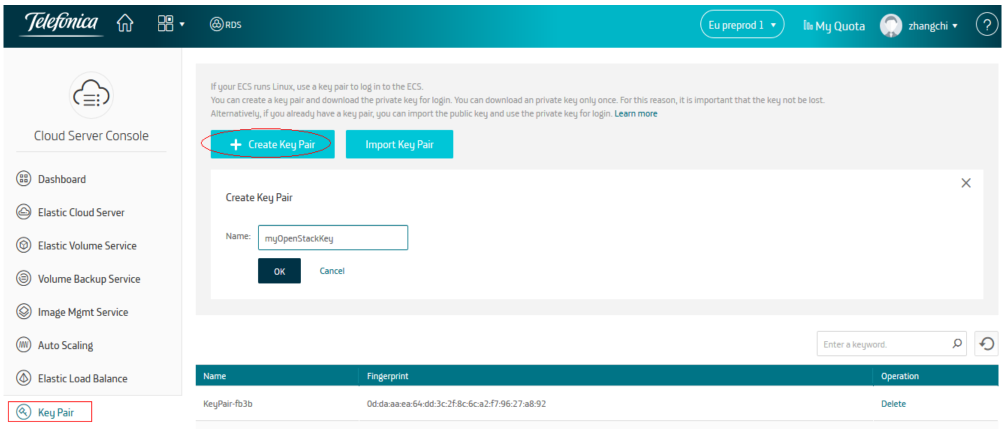
You can consume messages using the Python OpenStack SDK based on the following code:

```
msgList =  conn.dms.consume_message(queue, group[0].id)
```

2.4 SDK-related FAQ

2.4.1 How Can I Create a Key Pair on the Console?

Click **Create SSH Key Pair** to create a key pair **myOpenStackKey**, click **OK**, and save the **myOpenStackKey.pem** file to the local PC.



If your ECS runs Linux, use a key pair to log in to the ECS.
You can create a key pair and download the private key for login. You can download an private key only once. For this reason, it is important that the key not be lost.
Alternatively, if you already have a key pair, you can import the public key and use the private key for login. Learn more

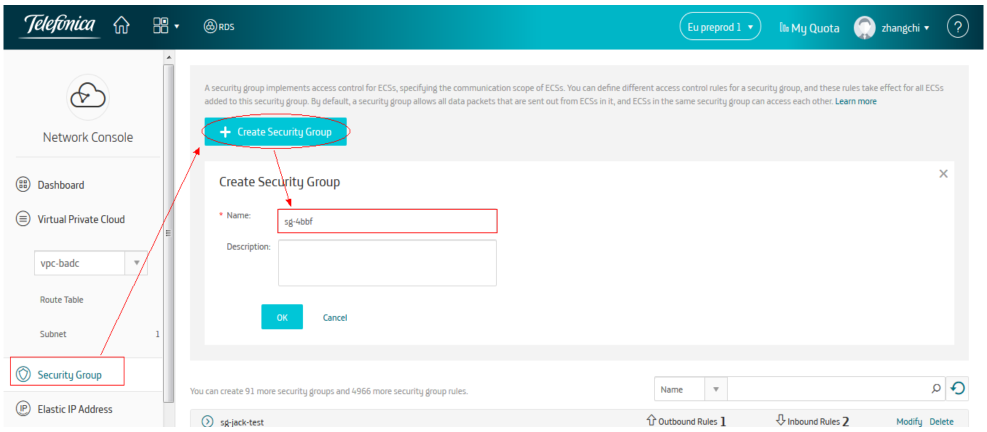
Create Key Pair

Name: myOpenStackKey

Name	Fingerprint	Operation
KeyPair-fb3b	0d:da:aa:ea:64:dd:3c:2f:8c:6c:a2:f7:96:27:a8:92	Delete

2.4.2 How Can I Create a Security Group?

1. Choose **Security Group** > **Create Security Group**.

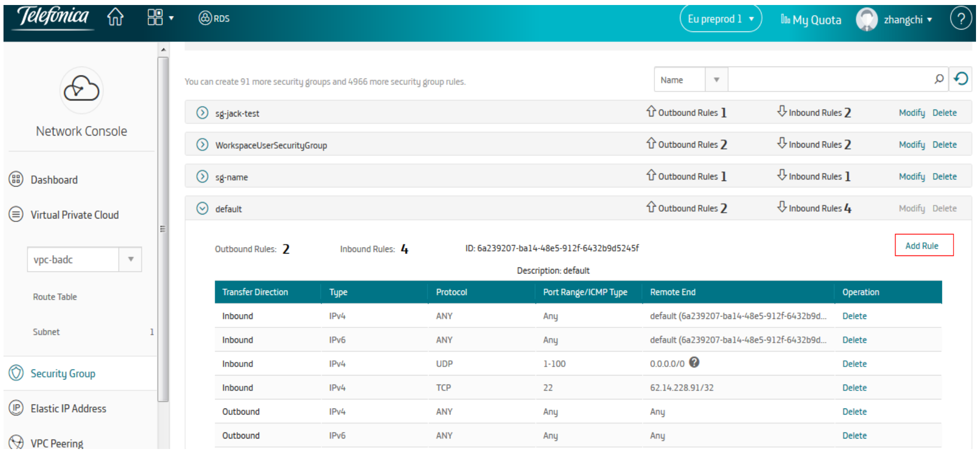

A security group implements access control for ECSs, specifying the communication scope of ECSs. You can define different access control rules for a security group, and these rules take effect for all ECSs added to this security group. By default, a security group allows all data packets that are sent out from ECSs in it, and ECSs in the same security group can access each other. Learn more

Create Security Group

* Name:	sg-4bbf
Description:	

You can create 91 more security groups and 4966 more security group rules.

2. Click **Add Rule**. On the displayed dialog box, add rules.


You can create 91 more security groups and 4966 more security group rules.

sg-jack-test	Outbound Rules 1	Inbound Rules 2	Modify Delete
WorkspaceUserSecurityGroup	Outbound Rules 2	Inbound Rules 2	Modify Delete
sg-name	Outbound Rules 1	Inbound Rules 1	Modify Delete
default	Outbound Rules 2	Inbound Rules 4	Modify Delete

Outbound Rules: 2 Inbound Rules: 4 ID: 6a239207-ba14-48e5-912f-6432b9d5245f

Description: default

Transfer Direction	Type	Protocol	Port Range/ICMP Type	Remote End	Operation
Inbound	IPv4	ANY	Any	default (6a239207-ba14-48e5-912f-6432b9d...	Delete
Inbound	IPv6	ANY	Any	default (6a239207-ba14-48e5-912f-6432b9d...	Delete
Inbound	IPv4	UDP	1-100	0.0.0.0/0	Delete
Inbound	IPv4	TCP	22	62.14.228.91/32	Delete
Outbound	IPv4	ANY	Any	Any	Delete
Outbound	IPv6	ANY	Any	Any	Delete

2.4.3 How Can I Obtain domain_name, project_name, and project_id?

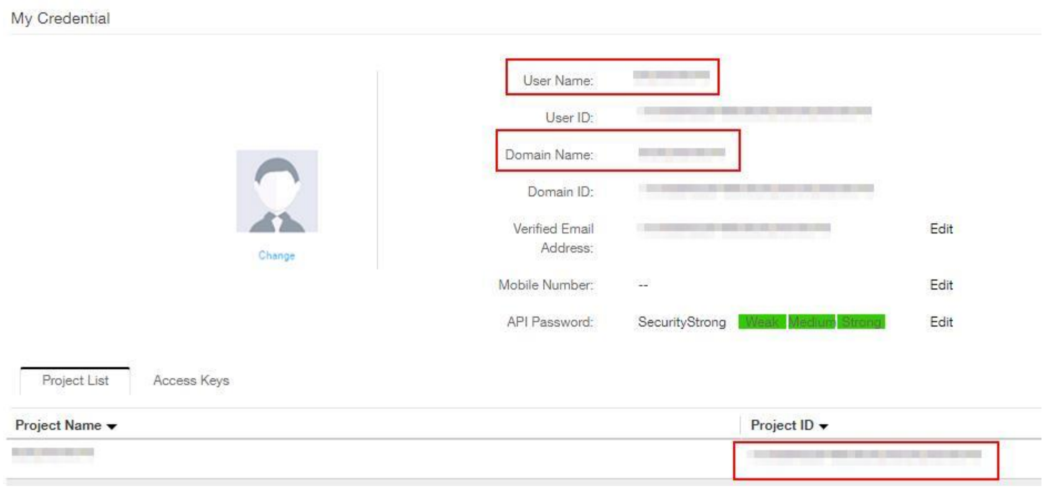
Prerequisites

You have logged in to the management console.

Procedure

Step 1 Click the username in the upper right corner and select **My Credential** from the drop-down menu.

Step 2 On the **My Credential** page, obtain the username, domain name, and project ID.


My Credential

Change

User Name:	[redacted]	
User ID:	[redacted]	
Domain Name:	[redacted]	
Domain ID:	[redacted]	
Verified Email Address:	[redacted]	Edit
Mobile Number:	--	Edit
API Password:	SecurityStrong Weak Medium Strong	Edit

Project List   Access Keys

Project Name	Project ID
[redacted]	[redacted]

----End

A Mapping Between API and SDK

A.1 Java

A.1.1 IAM

The SDK interfaces based on the Keystone v3 API are as follows. For details about the invoking methods, see the sample codes.

Interface	Method	API
ProjectService	List<? extends Project> list()	GET /v3/projects
ServiceEndpointService	List<? extends Service> list()	GET /v3/services
	List<? extends Endpoint> listEndpoints()	GET /v3/endpoints
TokenService	Token get(String tokenId)	GET /v3/auth/tokens
UserService	User create(String domainId, String name, String password, String email, boolean enabled)	POST /v3/users
	User create(User user)	POST /v3/users
	ActionResponse delete(String userId)	DELETE /v3/users/{user_id}
	User get(String userId)	GET /v3/users/{user_id}
	List<? extends User> getByName(String userName)	GET /v3/users?name={user_name}
	User getByName(String userName, String domainId)	GET /v3/users?name={user_name} &&domain_id={domain_id}
	List<? extends User> list()	GET /v3/users
	List<? extends Group> listUserGroups(String userId)	GET /v3/users/{user_id}/groups
	List<? extends Project>	GET

Interface	Method	API
	listUserProjects(String userId)	/v3/users/{user_id}/projects
	User update(User user)	PATCH /v3/users/{user_id}

A.1.2 IMS

The SDK interfaces based on the Glance v2 API are as follows. For details about the invoking methods, see the sample codes.

Interface	Method	API
ImageService	Image create(Image image)	POST /v2/images
	ActionResponse upload(String imageID,Payload payload,Image image)	PUT /v2/images/{image_id}/file
	ActionResponse Delete(String imageID)	DELETE /v2/images/{image_id}
	List<? extends Image> list()	GET /v2/images
	Image get(String imageID)	GET /v2/images/{image_id}
	ActionResponse updateTag(String imageID,String tagkeyvalue)	PUT /v2/images/{image_id}/tags/{tag}
	ActionResponse deleteTag(String tagkey,String tagvalue)	DELETE /v2/images/{image_id}/tags/{tag}
	List<? extends Member> listMembers(String imageid)	GET /v2/images/{image_id}/members
	Member getMember(String imageID,"memberid")	GET /v2/images/{image_id}/members/{member_id}
	ActionResponse deleteMember(String imageId,String memberID)	DELETE /v2/images/{image_id}/members/{member_id}
	Member updateMember(String imageid, String memberid, Member.MemberStatus.ACCEPTED));	PUT /v2/images/{image_id}/members/{member_id}

A.1.3 VPC

The SDK interfaces based on the Neutron v2.0 API are as follows. For details about the invoking methods, see the sample codes.

Interface	Method	API
NetFloatingIPService	NetFloatingIP associateToPort(String id, String portId)	PUT /v2.0/floatingips/{floatingip-id}
	NetFloatingIP create(NetFloatingIP floatingIp)	POST /v2.0/floatingips
	ActionResponse delete(String id)	DELETE /v2.0/floatingips/{floatingip-id}
	NetFloatingIP disassociateFromPort(String id)	PUT /v2.0/floatingips/{floatingip-id}
	NetFloatingIP get(String id)	GET /v2.0/floatingips/{floatingip-id}
	List<? extends NetFloatingIP> list()	GET /v2.0/floatingips
	List<? extends NetFloatingIP> list(Map<String,String> filteringParams)	GET /v2.0/floatingips
NetworkService	Network create(Network network)	post /v2.0/networks
	ActionResponse delete(String networkId)	DELETE /v2.0/networks/{network_id }
	Network get(String networkId)	GET /v2.0/networks/{network_id }
	List<? extends Network> list()	GET /v2.0/networks
	Network update(String networkId, NetworkUpdate network)	PUT /v2.0/networks/{network_id }
PortService	Port create(Port port)	POST /v2.0/ports
	ActionResponse delete(String portId)	DELETE /v2.0/ports/{port_id}
	Port get(String portId)	GET /v2.0/ports/{port_id}

Interface	Method	API
	List<? extends Port> list()	GET /v2.0/ports
	List<? extends Port> list(PortListOptions options)	GET /v2.0/ports?network_id={network_id}
	Port update(Port port)	PUT /v2.0/ports/{port_id}
RouterService	RouterInterface attachInterface(String routerId, AttachInterfaceType type, String portOrSubnetId)	PUT /v2.0/routers/{router_id}/add_router_interface
	Router create(Router router)	POST /v2.0/routers
	Router create(String name, boolean adminStateUp)	POST /v2.0/routers
	ActionResponse delete(String routerId)	DELETE /v2.0/routers/{router_id}
	RouterInterface detachInterface(String routerId, String subnetId, String portId)	PUT /v2.0/routers/{router_id}/remove_router_interface
	Router get(String routerId)	GET /v2.0/routers/{router_id}
	List<? extends Router>list()	GET /v2.0/routers
	Router toggleAdminStateUp(String routerId, boolean adminStateUp)	PUT /v2.0/routers/{router_id}
	Router update(Router router)	PUT /v2.0/routers/{router_id}
SecurityGroupRuleService	SecurityGroupRule create(SecurityGroupRule rule)	POST /v2.0/security-group-rules
	void delete(String id)	DELETE /v2.0/security-group-rules/{security-group-rules-id}
	SecurityGroupRule get(String id)	GET /v2.0/security-group-rules/{security-group-rules-id}
	List<? extends SecurityGroupRule> list()	GET /v2.0/security-group-rules
SecurityGroupService	SecurityGroup create(SecurityGroup	POST /v2.0/security-groups

Interface	Method	API
	securityGroup)	
	ActionResponse delete(String id)	DELETE /v2.0/security-groups/{security-group-id}
	SecurityGroup get(String id)	GET /v2.0/security-groups/{security-group-id}
	List<? extends SecurityGroup>list()	GET /v2.0/security-groups
SubnetService	Subnet create(Subnet subnet)	POST /v2.0/subnets
	ActionResponse delete(String subnetId)	DELETE /v2.0/subnets/{subnet_id}
	Subnet get(String subnetId)	GET /v2.0/subnets/{subnet_id}
	List<? extends Subnet>list()	GET /v2.0/subnets
	Subnet update(String subnetId, Subnet subnet)	PUT /v2.0/subnets/{subnet_id}
	Subnet update(Subnet subnet)	PUT /v2.0/subnets/{subnet_id}

The SDK interfaces based on the VPC v1 API are as follows. For details about the invoking methods, see the sample codes.

Interface	Method	API
PublicIpService	VirtualPublicIp get(String publicipId)	GET /v1/{tenant_id}/publicips/{publicip_id}
BandWidthService	VirtualBandWidths get(String bandwidthId)	GET /v1/{tenant_id}/bandwidths/{bandwidth_id}
	List<VirtualBandWidths> list()	GET /v1/{tenant_id}/bandwidths
	List<VirtualBandWidths> list(Map<String, String> filteringParams)	GET /v1/{tenant_id}/bandwidths

The SDK interfaces based on the VPC v2.0 API are as follows. For details about the invoking methods, see the sample codes.

Interface	Method	API

Interface	Method	API
PublicIpService	AsyncPublicipRespEntity apply(VirtualPublicIps virtualPublicIps)	POST /v2.0/{tenant_id}/publicips
BandWidthService	AsyncBandWidthRespEntity update(VirtualBandWidths bandWidth,String bandwidthId)	PUT / v2.0/{tenant_id}/bandwidths/{bandwidth_id}

A.1.4 ECS

The SDK interfaces based on the Nova v2 API are as follows. For details about the invoking methods, see the sample codes.

Interface	Method	API
ComputeFloatingIPService	ActionResponse addFloatingIP(Server server, String ipAddress)	POST /v2/{tenant_id}/servers/{server_id}/action
	ActionResponse addFloatingIP(Server server, String fixedIpAddress, String ipAddress)	POST /v2/{tenant_id}/servers/{server_id}/action
	FloatingIP allocateIP(String pool)	POST /v2/{tenant_id}/os-floating-ips
	ActionResponse deallocateIP(String id)	DELETE /v2/{tenant_id}/os-floating-ips/{floating_ip_id}
	List<? extends FloatingIP> list()	GET /v2/{tenant_id}/os-floating-ips
	ActionResponse removeFloatingIP(Server server, String ipAddress)	POST /v2/{tenant_id}/servers/{server_id}/action
ComputeImageService	ActionResponse delete(String imageId)	DELETE /v2/{tenant_id}/images/{image_id}
	Image get(String imageId)	GET /v2/{tenant_id}/images/{image_id}
	List<? extends Image> list()	GET /v2/{tenant_id}/images/detail
	List<? extends Image> list(boolean	GET

Interface	Method	API
	detailed)	/v2/{tenant_id}/images
ComputeSecurityGroupService	SecGroupExtension create(String name, String description)	POST /v2/{tenant_id}/os-security-groups
	SecGroupExtension.Rule createRule(SecGroupExtension.Rule rule)	POST /v2/{tenant_id}/os-security-group-rules
	ActionResponse delete(String securityGroupId)	DELETE /v2/{tenant_id}/os-security-groups/{security_group}
	ActionResponse deleteRule(String ruleId)	DELETE /v2/{tenant_id}/os-security-group-rules/{security_group_rule_id}
	SecGroupExtension get(String securityGroupId)	GET /v2/{tenant_id}/os-security-groups/{security_group_id}
	List<? extends SecGroupExtension> list()	GET /v2/{tenant_id}/os-security-groups
FlavorService	Flavor get(String flavorId)	GET /v2/{tenant_id}/flavors/{flavor_id}
	List<? extends Flavor> list()	GET /v2/{tenant_id}/flavors/detail?is_public=None
KeypairService	Keypair create(String name, String publicKey)	POST /v2/{tenant_id}/os-keypairs
	ActionResponse delete(String name)	DELETE /v2/{tenant_id}/os-keypairs/{keypair_name}
	Keypair get(String name)	GET /v2/{tenant_id}/os-keypairs/{keypair_name}
	List<? extends Keypair> list()	GET /v2/{tenant_id}/os-keypairs
QuotaSetService	QuotaSet get(String tenantId)	GET /v2/{tenant_id}/os-quota-sets/{tenant_id}
	Limits limits()	GET /v2/{tenant_id}/limits
ServerGroupService	ServerGroup create(String name,	POST

Interface	Method	API
	String policy)	/v2/{tenant_id}/os-server-groups
	ActionResponse delete(String id)	DELETE /v2/{tenant_id}/os-server-groups/{{server_group_id}}
	ServerGroup get(String id)	GET /v2/{tenant_id}/os-server-groups/{server_group_id}
	List<? extends ServerGroup> list()	GET /v2/{tenant_id}/os-server-groups
ServerService	ActionResponse action(String serverId, Action action) Executes the specified Action such as RESUME, PAUSE, START, and STOP.	POST /v2/{tenant_id}/servers/{server_id}/action
	VolumeAttachment attachVolume(String serverId, String volumeId, String device)	POST /v2/{tenant_id}/servers/{server_id}/os-volume_attachments
	Server boot(ServerCreate server)	POST /v2/{tenant_id}/servers
	Server bootAndWaitActive(ServerCreate server, int maxWaitTime)	POST /v2/{tenant_id}/servers GET /v2/{tenant_id}/servers/{server_id}
	ActionResponse confirmResize(String serverId)	POST /v2/{tenant_id}/servers/{server_id}/action
	String createSnapshot(String serverId, String snapshotName)	POST /v2/{tenant_id}/servers/{server_id}/action
	ActionResponse delete(String serverId)	DELETE /v2/{tenant_id}/servers/{server_id}
	ActionResponse deleteMetadataItem(String serverId, String key)	DELETE /v2/{tenant_id}/servers/{server_id}/metadata/{key}
	ActionResponse detachVolume(String serverId, String attachmentId)	DELETE /v2/{tenant_id}/servers/{server_id}/os-volume_attachments/{attachment_id}

Interface	Method	API
	Server get(String serverId)	GET /v2/{tenant_id}/servers/{server_id}
	Map<String,String> getMetadata(String serverId)	GET /v2/{tenant_id}/servers/{server_id}/metadata
	List<? extends Server> list()	GET /v2/{tenant_id}/servers/detail
	List<? extends Server> list(boolean detail)	GET /v2/{tenant_id}/servers
	List<? extends Server> list(Map<String,String> filteringParams)	GET /v2/{tenant_id}/servers/detail{?changes-since,image,flavor,name,status,host,limit,marker}
	ActionResponse reboot(String serverId, RebootType type)	POST /v2/{tenant_id}/servers/{server_id}/action
	ActionResponse rebuild(String serverId, RebuildOptions options)	POST /v2/{tenant_id}/servers/{server_id}/action
	ActionResponse resize(String serverId, String flavorId)	POST /v2/{tenant_id}/servers/{server_id}/action
	ActionResponse revertResize(String serverId)	POST /v2/{tenant_id}/servers/{server_id}/action
	Server update(String serverId, ServerUpdateOptions options)	PUT /v2/{tenant_id}/servers/{server_id}
	Map<String,String> updateMetadata(String serverId, Map<String,String> metadata)	PUT /v2/{tenant_id}/servers/{server_id}/metadata
	Server waitForServerStatus(String serverId, Server.Status status, int maxWait, TimeUnit maxWaitUnit)	GET /v2/{tenant_id}/servers/{server_id}
InterfaceService (ext)	InterfaceAttachment create(String serverId, String portId)	POST /v2/{tenant_id}/servers/{server_id}/os-interface
	ActionResponse detach(String serverId, String attachmentId)	DELETE /v2/{tenant_id}/servers/{server_id}/os-interface/{port

Interface	Method	API
		_id}
	InterfaceAttachment get(String serverId, String attachmentId)	GET /v2/{tenant_id}/servers/{server_id}/os-interface/{port_id}
	List<? extends InterfaceAttachment> list(String serverId)	GET /v2/{tenant_id}/servers/{server_id}/os-interface
ZoneService(ext)	List<? extends AvailabilityZone> list()	GET /v2/{tenant_id}/os-availability-zone

The SDK interfaces based on the ECS v1 API are as follows. For details about the invoking methods, see the sample codes.

Interface	Method	API
CloudServerService	String create(ServerCreate creation)	POST /v1/{tenant_id}/cloudservers
	List<CloudServer> list()	GET /v1/{tenant_id}/cloudservers/detail
	CloudServer get(String serverId)	GET /v1/{tenant_id}/cloudservers/{server_id}
	String resize(ResizeServer resize,String serverId)	POST /v1/{tenant_id}/cloudservers/{server_id}/resize
	String delete(List<String> serverIds, boolean deletePublicIp, boolean deleteVolume)	POST /v1/{tenant_id}/cloudservers/action
	String stop(List<String> serverIds, StopType type)	POST /v1/{tenant_id}/cloudservers/action
	String reboot(List<String> serverIds, RebootType type)	POST /v1/{tenant_id}/cloudservers/action
	String start(List<String> serverIds)	POST /v1/{tenant_id}/cloudservers/action
JobService	Job get(String jobId)	GET /v1/{tenant_id}/jobs/{job_i

Interface	Method	API
		d}

The SDK interfaces based on the ECS v1.1 API are as follows. For details about the invoking methods, see the sample codes.

Interface	Method	API
CloudServerService	AsyncRespEntity create(ServerCreate creation)	POST /v1.1/{tenant_id}/cloudservers
	AsyncRespEntity resize(ResizeServer resize,String serverId)	POST /v1.1/{tenant_id}/cloudservers/{server_id}/resize

A.1.5 EVS

The SDK interfaces based on the Cinder v2 API are as follows. For details about the invoking methods, see the sample codes.

Interface	Method	API
BlockVolumeService	Volume create(Volume volume)	POST /v2/{tenant_id}/volumes
	ActionResponse delete(String volumeId)	DELETE /v2/{tenant_id}/volumes/{volume_id}
	ActionResponse extend(String volumeId, Integer newSize)	POST /v2/{tenant_id}/volumes/{volume_id}/action
	Volume get(String volumeId)	GET /v2/{tenant_id}/volumes/{volume_id}
	List<? extends Volume> list()	GET /v2/{tenant_id}/volumes/detail
	List<? extends Volume> list(Map<String,String> filteringParams)	GET /v2/{tenant_id}/volumes/detail?limit={limit_nmuber} GET /v2/{tenant_id}/volumes/detail?marker={volume_id}
	ActionResponse update(String volumeId,	PUT /v2/{tenant_id}/volumes/{

Interface	Method	API
	String name, String description)	volume_id}
BlockVolumeSnapshotService	ActionResponse delete(String snapshotId)	DELETE /v2/{tenant_id}/snapshots/{snapshot_id}
	VolumeSnapshot get(String snapshotId)	GET /v2/{tenant_id}/snapshots/{snapshot_id}
	List<? extends VolumeSnapshot> list()	GET /v2/{tenant_id}/snapshots
	List<? extends VolumeSnapshot> list(Map<String,String> filteringParams)	GET /v2/{tenant_id}/snapshots ?volume_id={volume_id}
CinderZoneService	List<? extends AvailabilityZone> list()	GET /v2/{tenant_id}/os-availab ility-zone

The SDK interfaces based on the EVS v2.1 API are as follows. For details about the invoking methods, see the sample codes.

Interface	Method	API
VolumeService	AsyncRespEntity create(Volumes volume)	POST /v2.1/{tenant_id}/cloudvol umes
	AsyncRespEntity extend(Extend extend,String volumeId)	POST /v2.1/{tenant_id}/cloudvol umes/{volume_id}/action

A.1.6 RTS

The SDK interfaces based on the Heat v1 API are as follows. For details about the invoking methods, see the sample codes.

Interface	Method	API
EventsService	List<? extends Event> list(String stackName, String stackId)	GET /V1/{tenant_id}/stacks/{stack _name}/{stack_id}/events
	List<? extends Event> list(String stackName, String stackId, String resourceName)	GET /V1/{tenant_id}/stacks/{stack _name}/{stack_id}/resources/{resource_name}/events

Interface	Method	API
	Event show(String stackName, String stackId, String resourceName, String eventId)	GET /V1/{tenant_id}/stacks/{stack_name}/{stack_id}/resources/{resource_name}/events/{event_id}
ResourcesService	List<? extends Resource> list(String stackNameOrId)	GET /V1/{tenant_id}/stacks/{stack_name}/{stack_id}/resources
	List<? extends Resource> list(String stackName, String stackId)	GET /V1/{tenant_id}/stacks/{stack_name}/{stack_id}/resources
	Resource show(String stackName, String stackId, String resourceName)	GET /V1/{tenant_id}/stacks/{stack_name}/{stack_id}/resources/{resource_name}
SoftwareConfigServic	SoftwareConfig create(SoftwareConfig sc)	POST /V1/{tenant_id}/software_configs
	ActionResponse delete(String configId)	DELETE /V1/{tenant_id}/software_configs/{software_config_id}
	SoftwareConfig show(String configId)	GET /V1/{tenant_id}/software_configs/{software_config_id}
StackService	Stack create(StackCreate newStack)	POST /V1/{tenant_id}/stacks
	Stack create(String name, String template, Map<String,String> parameters, boolean disableRollback, Long timeOutMins)	POST /V1/{tenant_id}/stacks
	ActionResponse delete(String stackName, String stackId)	DELETE /V1/{tenant_id}/stacks/{stack_name}/{stack_id}
	Stack getDetails(String stackName, String stackId)	GET /V1/{tenant_id}/stacks/{stack_name}/{stack_id}
	Stack getStackByName(String name)	GET /V1/{tenant_id}/stacks/{stack_name}/{stack_id}
	List<? extends Stack> list()	GET /V1/{tenant_id}/stacks
	ActionResponse update(String	PUT

Interface	Method	API
	stackName, String stackId, StackUpdate stackUpdate)	/V1/{tenant_id}/stacks/{stack_name}/{stack_id}
TemplateService	Map<String,Object> getTemplateAsMap(String stackNameOrId)	GET /V1/{tenant_id}/stacks/{stack_name}/template
	Map<String,Object> getTemplateAsMap(String stackName, String stackId)	GET /V1/{tenant_id}/stacks/{stack_name}/{stack_id}/template
	String getTemplateAsString(String stackName, String stackId)	GET /V1/{tenant_id}/stacks/{stack_name}/{stack_id}/template
	TemplateResponse validateTemplate(String template)	POST /V1/{tenant_id}/validate
	TemplateResponse validateTemplate(Template template)	POST /V1/{tenant_id}/validate
	TemplateResponse validateTemplateByURL(String templateURL)	POST /V1/{tenant_id}/validate

A.1.7 AS

The SDK interfaces based on the AS v1 API are as follows. For details about the invoking methods, see the sample codes.

Interface	Method	API
Group	String create(ScalingGroupCreate group)	POST /autoscaling-api/v1/{tenant_id}/scaling_group
	List<? extends ScalingGroup> list()	GET /autoscaling-api/v1/{tenant_id}/scaling_group
	ScalingGroup get(String groupId)	GET /autoscaling-api/v1/{tenant_id}/scaling_group/{scaling_group_id}
	String update(String groupId, ScalingGroupUpdate group)	PUT /autoscaling-api/v1/{tenant_id}/scaling_group/{scaling_group_id}
	ActionResponse delete(String groupId)	DELETE /autoscaling-api/v1/{tenant_id}/scaling_group/{scaling_group_id}

Interface	Method	API
	ActionResponse resume(String groupId)	POST /autoscaling-api/v1/{tenant_id}/scaling_group/{scaling_group_id}/action
	ActionResponse pause(String groupId)	POST /autoscaling-api/v1/{tenant_id}/scaling_group/{scaling_group_id}/action
config	String create(ScalingConfigCreate config)	POST /autoscaling-api/v1/{tenant_id}/scaling_configuration
	List<? extends ScalingConfig> list()	GET /autoscaling-api/v1/{tenant_id}/scaling_configuration
	ScalingConfig get(String configId)	GET /autoscaling-api/v1/{tenant_id}/scaling_configuration/{scaling_configuration_id}
	ActionResponse delete(String configId)	DELETE /autoscaling-api/v1/{tenant_id}/scaling_configuration/{scaling_configuration_id}
	ActionResponse delete(List<String> configIds)	POST /autoscaling-api/v1/{tenant_id}/scaling_configurations
Instance	List<? extends ScalingGroupInstance> list(String groupId, ScalingGroupInstanceListOptions options)	DELETE /autoscaling-api/v1/{tenant_id}/scaling_group_instance/{instance_id}
	ActionResponse delete(String instanceId, boolean deleteInstance)	POST /autoscaling-api/v1/{tenant_id}/scaling_group_instance/{scaling_group_id}/action
	ActionResponse batchAdd(String groupId, List<String> instanceIds, boolean deleteInstance)	POST /autoscaling-api/v1/{tenant_id}/scaling_group_instance/{scaling_group_id}/action
	ActionResponse batchRemove(String groupId, List<String> instanceIds, boolean deleteInstance)	POST /autoscaling-api/v1/{tenant_id}/scaling_group_instance/{scaling_group_id}/action
Policy	String create(ScalingPolicyCreateUpdate policy)	POST /autoscaling-api/v1/{tenant_id}/scaling_policy

Interface	Method	API
	String update(ScalingPolicyCreateUpdate policy)	PUT /autoscaling-api/v1/{tenant_id}/scaling_policy/{scaling_policy_id}
	List<? extends ScalingPolicy> list(String groupId)	GET /autoscaling-api/v1/{tenant_id}/scaling_policy/{scaling_group_id}/list
	ScalingPolicy get(String policyId)	GET /autoscaling-api/v1/{tenant_id}/scaling_policy/{scaling_policy_id}
	ActionResponse execute(String policyId)	POST /autoscaling-api/v1/{tenant_id}/scaling_policy/{scaling_policy_id}/action
	ActionResponse resume(String policyId)	POST /autoscaling-api/v1/{tenant_id}/scaling_policy/{scaling_policy_id}/action
	ActionResponse pause(String policyId)	POST /autoscaling-api/v1/{tenant_id}/scaling_policy/{scaling_policy_id}/action
	ActionResponse delete(String policyId)	DELETE /autoscaling-api/v1/{tenant_id}/scaling_policy/{scaling_policy_id}
Activity	list(String groupId, ScalingActivityLogListOptions options)	GET /autoscaling-api/v1/{tenant_id}/scaling_activity_log/{scaling_group_id}
Quota	List<Quota> list()	GET /autoscaling-api/v1/{tenant_id}/quotas
	List<Quota> list(String groupId)	GET /autoscaling-api/v1/{tenant_id}/quotas/{scaling_group_id}
Lifecycle Hook	ASAutoScalingLifecycleHook create(ASAutoScalingLifecycleHook lifecycleHook , String groupId)	POST /autoscaling-api/v1/{tenant_id}/scaling_lifecycle_hook/{scaling_group_id}
	List<? extends ASAutoScalingLifecycleHook> list(String groupId)	GET /autoscaling-api/v1/{tenant_id}/scaling_lifecycle_hook/{scaling_g

Interface	Method	API
		roup_id}/list
	ASAutoScalingLifecycleHook list(String groupId , String lifecycleHookName)	GET /autoscaling-api/v1/{tenant_id}/scaling_lifecycle_hook/{scaling_group_id}/{lifecycle_hook_name}
	ActionResponse delete(String groupId , String lifecycleHookName)	DELETE /autoscaling-api/v1/{tenant_id}/scaling_lifecycle_hook/{scaling_group_id}/{lifecycle_hook_name}
	ASAutoScalingLifecycleHook update(String groupId , String lifecycleHookName , ASAutoScalingLifecycleHook lifecycleHook)	PUT /autoscaling-api/v1/{tenant_id}/scaling_lifecycle_hook/{scaling_group_id}/{lifecycle_hook_name}
	List<? extends AutoScalingInstanceHangupInfo> scalingInstanceHangup(String groupId , ScalingInstanceOptions instanceId)	GET /autoscaling-api/v1/{tenant_id}/scaling_instance_hook/{scaling_group_id}/list
	ActionResponse scalingInstanceHookCallback(String groupId , ASAutoScalingLifecycleInstanceCallback lifecycleInstanceCallback)	PUT /autoscaling-api/v1/{tenant_id}/scaling_instance_hook/{scaling_group_id}/callback

A.1.8 CES

The SDK interfaces based on the CES v1.0 API are as follows. For details about the invoking methods, see the sample codes.

Interface	Method	API
MetricService	List<? extends Metric> getList(MetricFilterOptions options);	GET /V1.0/{project_id}/metrics
AlarmService	List<? extends Alarm> list(AlarmFilterOptions options);	GET /V1.0/{project_id}/alarms
	List<? extends Alarm> get(String alarmId);	GET /V1.0/{project_id}/alarms/{alarm_id}
	ActionResponse startAlarm(String alarmId)	PUT /V1.0/{project_id}/alarms/{alarm_id}/action
	ActionResponse deleteAlarm(String alarmId);	DELETE /V1.0/{project_id}/alarms/{

Interface	Method	API
		alarm_id}
MetricDataService	MetricAggregation get(String namespace, String metric_name, Date from, Date to, Period period, Filter filter, String[] dimValues);	GET /V1.0/{project_id}/metric-data
	ActionResponse add(List<? extends MetricData> metrics);	POST /V1.0/{project_id}/metric-data
QuotaService	CloudEyeQuota get();	GET /V1.0/{project_id}/quotas

A.1.9 DNS

The SDK interfaces based on the DNS v2 API are as follows. For details about the invoking methods, see the sample codes.

Interface	Method	API
Zone	osclient.dns().zones().create(zone)	POST /v2/zones
	osclient.dns().zones().get("zone-id")	GET /v2/zones/{zone_id}
	osclient.dns().zones().list()	GET /v2/zones
	osclient.dns().zones().delete(zone_id)	DELETE /v2/zones/{zone_id}
	osclient.dns().zones().listNameservers(zone_id)	GET /v2/zones/{zone_id}/nameservers
	osclient.dns().zones().associateRouter(zone_id, router)	POST /v2/zones/{zone_id}/associaterouter
	osclient.dns().zones().disassociateRouter(zone_id, router)	POST /v2/zones/{zone_id}/disassociaterouter
Recordset	osclient.dns().recordsets().create(ZONE_ID, recordset)	POST /v2/zones/{zone_id}/recordsets
	osclient.dns().recordsets().get(zone_id, recordset_id)	GET /v2/zone/{zone_id}/recordsets/{recordset_id}
	osclient.dns().recordsets().list()	GET /v2/recordsets
	osclient.dns().recordsets().list(zone_id)	GET /v2/zones/{zone_id}/recordsets
	osclient.dns().recordsets().delete(zone_id, recordset_id)	DELETE /v2/zones/{zone_id}/recordsets/

Interface	Method	API
		{recordset_id}
PTR Record	osclient.dns().ptrs().setup(ptrRecord)	PATCH /v2/reverse/floatingips/{region}:{floatingip_id}
	osclient.dns().ptrs().restore(region, floatingIpId)	PATCH /v2/reverse/floatingips/{region}:{floatingip_id}
	osclient.dns().ptrs().list()	GET /v2/reverse/floatingips
	osclient.dns().ptrs().get(region, floatingIpId)	GET /v2/reverse/floatingips/{region}:{floatingip_id}

A.1.10 ELB

The SDK interfaces based on the ELB v1.0 API are as follows. For details about the invoking methods, see the sample codes.

Interface	Method	API
LoadBalancer	ELBJob create(LoadBalancerCreate loadBalancer)	POST /v1.0/{tenant_id}/elbaas/loadbalancers
	LoadBalancer get(String loadBalancerId)	GET /v1.0/{tenant_id}/elbaas/loadbalancers/{loadbalancer_id}
	List<? extends LoadBalancer> list()	GET /v1.0/{tenant_id}/elbaas/loadbalancers
	ELBJob update(String loadBalancerId, LoadBalancerUpdate loadBalancer)	PUT /v1.0/{tenant_id}/elbaas/loadbalancers/{loadbalancer_id}
	ELBJob delete(String loadBalancerId)	DELETE /v1.0/{tenant_id}/elbaas/loadbalancers/{loadbalancer_id}
Listener	ListenerCreate create(ListenerCreate listener)	POST /v1.0/{tenant_id}/elbaas/listeners
	Listener get(String listenerId)	GET /v1.0/{tenant_id}/elbaas/listeners/{listener_id}
	Listener[] list()	GET /v1.0/{tenant_id}/elbaas/listeners?loadbalancer_id={loadbalancer_id}

Interface	Method	API
	Listener update(String listenerId, ListenerUpdate listener)	PUT /v1.0/{tenant_id}/elbaas/listeners/{listener_id}
	ActionResponse delete(String listenerId)	DELETE /v1.0/{tenant_id}/elbaas/listeners/{listener_id}
HealthCheck	HealthCheck create(HealthCheckCreate healthCheck)	POST /v1.0/{tenant_id}/elbaas/healthcheck
	HealthCheck get(String healthCheckId)	GET /v1.0/{tenant_id}/elbaas/healthcheck/{healthcheck_id}
	HealthCheck update(String healthCheckId, HealthCheckUpdate healthCheck)	PUT /v1.0/{tenant_id}/elbaas/healthcheck/{healthcheck_id}
	ActionResponse delete(String healthCheckId)	DELETE /v1.0/{tenant_id}/elbaas/healthcheck/{healthcheck_id}
Member	ELBJob create(String listenerId, List<ServerCreate> servers)	POST /v1.0/{tenant_id}/elbaas/listeners/{listener_id}/members
	ELBJob delete(String listenerId, ServerDelete serverDelete)	POST /v1.0/{tenant_id}/elbaas/listeners/{listener_id}/members/action
	Server[] list(String listenerId)	GET /v1.0/{tenant_id}/elbaas/listeners/{listener_id}/members
Certificate	Certificate create(Certificate cert)	POST /v1.0/{tenant_id}/elbaas/certificate
	Certificates list()	GET /v1.0/{tenant_id}/elbaas/certificate
	Certificate update(String certificateId, CertificateUpdate cert)	PUT /v1.0/{tenant_id}/elbaas/certificate/{certificate_id}
	ActionResponse delete(String certificateId)	DELETE /v1.0/{tenant_id}/elbaas/certificate/{certificate_id}

A.1.11 VBS

The SDK interfaces based on the VBS v2 API are as follows. For details about the invoking methods, see the sample codes.

Interface	Method	API
LoadBalancer	ELBJob create(LoadBalancerCreate loadBalancer)	POST /v1.0/{tenant_id}/elbaas/loadbalancers
	LoadBalancer get(String loadBalancerId)	GET /v1.0/{tenant_id}/elbaas/loadbalancers/{loadbalancer_id}
	List<? extends LoadBalancer> list()	GET /v1.0/{tenant_id}/elbaas/loadbalancers
	ELBJob update(String loadBalancerId, LoadBalancerUpdate loadBalancer)	PUT /v1.0/{tenant_id}/elbaas/loadbalancers/{loadbalancer_id}
	ELBJob delete(String loadBalancerId)	DELETE /v1.0/{tenant_id}/elbaas/loadbalancers/{loadbalancer_id}
Listener	ListenerCreate create(ListenerCreate listener)	POST /v1.0/{tenant_id}/elbaas/listeners
	Listener get(String listenerId)	GET /v1.0/{tenant_id}/elbaas/listeners/{listener_id}
	Listener[] list()	GET /v1.0/{tenant_id}/elbaas/listeners?loadbalancer_id={loadbalancer_id}
	Listener update(String listenerId, ListenerUpdate listener)	PUT /v1.0/{tenant_id}/elbaas/listeners/{listener_id}
	ActionResponse delete(String listenerId)	DELETE /v1.0/{tenant_id}/elbaas/listeners/{listener_id}
HealthCheck	HealthCheck create(HealthCheckCreate healthCheck)	POST /v1.0/{tenant_id}/elbaas/healthcheck
	HealthCheck get(String healthCheckId)	GET /v1.0/{tenant_id}/elbaas/healthcheck/{healthcheck_id}
	HealthCheck update(String healthCheckId, HealthCheckUpdate healthCheck)	PUT /v1.0/{tenant_id}/elbaas/healthcheck/{healthcheck_id}

Interface	Method	API
	ActionResponse delete(String healthCheckId)	DELETE /v1.0/{tenant_id}/elbaas/healthcheck/{healthcheck_id}
Member	ELBJob create(String listenerId, List<ServerCreate> servers)	POST /v1.0/{tenant_id}/elbaas/listeners/{listener_id}/members
	ELBJob delete(String listenerId, ServerDelete serverDelete)	POST /v1.0/{tenant_id}/elbaas/listeners/{listener_id}/members/action
	Server[] list(String listenerId)	GET /v1.0/{tenant_id}/elbaas/listeners/{listener_id}/members
Certificate	Certificate create(Certificate cert)	POST /v1.0/{tenant_id}/elbaas/certificate
	Certificates list()	GET /v1.0/{tenant_id}/elbaas/certificate
	Certificate update(String certificateId, CertificateUpdate cert)	PUT /v1.0/{tenant_id}/elbaas/certificate/{certificate_id}
	ActionResponse delete(String certificateId)	DELETE /v1.0/{tenant_id}/elbaas/certificate/{certificate_id}

A.1.12 CTS

The SDK interfaces based on the CTS v1.0 API are as follows. For details about the invoking methods, see the sample codes.

Interface	method	API URL
Tracker	osclient.cloudTraceV1().trackers().create("bucket-name", "file-prefix")	POST /v1.0/{project_id}/tracker
	osclient.cloudTraceV1().trackers().get("tracker-name")	GET /v1.0/{project_id}/tracker{?tracker_name}
	osclient.cloudTraceV1().trackers().update(update)	PUT /v1.0/{project_id}/tracker/{?tracker_name}
	osclient.cloudTraceV1().trackers().delete("tracker-name")	DELETE /v1.0/{project_id}/tracker{?tracker_name}

Interface	method	API URL
Trace	osclient.cloudTraceV2().traces().list("system", options)	GET /v2.0/{project_id}/{tracker_name}/trace{?trace_id,service_type,resource_type,resource_id,resource_name,trace_name,trace_rating,user,limit,from,to,next}

A.1.13 SMN

The SDK interfaces based on the SMN v2 API are as follows. For details about the invoking methods, see the sample codes.

Interface	method	API URL
Topic	osclient.notification().topics().create("topic-name", "display-name")	POST /v2/{project_id}/notifications/topics
	osclient.notification().topics().updateDisplayName(topicUrn, displayName)	PUT /v2/{project_id}/notifications/topics/{topic_urn}
	osclient.notification().topics().delete(topicUrn)	DELETE /v2/{project_id}/notifications/topics/{topic_urn}
	osclient.notification().topics().list(100, 0)	GET /v2/{project_id}/notifications/topics?offset=0&limit=2
	osclient.notification().topics().get(topicUrn)	GET /v2/{project_id}/notifications/topics/{topic_urn}
	osclient.notification().topics().getTopicAttributes(topicUrn)	GET /v2/{project_id}/notifications/topics/{topic_urn}/attributes?name=access_policy
	osclient.notification().topics().getTopicAttribute(topicUrn, TopicAttributeName.Introduction)	/v2/{project_id}/notifications/topics/{topic_urn}/attributes?name=access_policy
	osclient.notification().topics().updateTopicAttribute(topicUrn, TopicAttributeName.Introduction, "sdk-unittest")	PUT /v2/{project_id}/notifications/topics/{topic_urn}/attributes/{attributes_name}
	osclient.notification().topics() .deleteTopicAttribute(topicUrn, TopicAttributeName.Introduction);	DELETE /v2/{project_id}/notifications/topics/{topic_urn}/attributes/{attributes_name}

Interface	method	API URL
	osclient.notification().topics().deleteTopicAttributes(topicUrn)	DELETE /v2/{project_id}/notifications/topics/{topic_urn}/attributes
Subscribe	osclient.notification().subscriptions().list(100, 0)	GET /v2/{project_id}/notifications/subscriptions?offset=0&limit=2
	osclient.notification().subscriptions().listByTopic("topic-urn", 100, 0)	GET /v2/{project_id}/notifications/topics/{topic_urn}/subscriptions?offset=0&limit=10
	osclient.notification().subscriptions().subscribe(subscribe)	POST /v2/{project_id}/notifications/topics/{topic_urn}/subscriptions
	osclient.notification().subscriptions().unsubscribe("subscription-urn")	DELETE /v2/{project_id}/notifications/subscriptions/{subscription_urn}
Message template	osclient.notification().messageTemplates().create(create)	POST /v2/{project_id}/notifications/message_template
	osclient.notification().messageTemplates().updateContent("message-template-id", "Hello, {user}")	PUT /v2/{project_id}/notifications/message_template/{message_template_Id}
	osclient.notification().messageTemplates().delete("message-template-id")	DELETE /v2/{project_id}/notifications/message_template/{message_template_id}
	osclient.notification().messageTemplates().list(options);	GET /v2/{project_id}/notifications/message_template
	osclient.notification().messageTemplates().get("message-template-id")	GET /v2/{project_id}/notifications/message_template/{message_template_id}
Message	osclient.notification().messages().publish("topic-urn", "subject", "message-content")	POST /v2/{project_id}/notifications/topics/{topic_urn}/publish
	osclient.notification().messages().publish("topic-urn", structuredMessage)	POST /v2/{project_id}/notifications/topics/{topic_urn}/publish

Interface	method	API URL
	osclient.notification().messages().publish("topic-urn", templatedMessage)	POST /v2/{project_id}/notifications/topics/{topic_urn}/publish
SMS	osclient.notification().sms().send("15659767757", "Hello, sms", null)	POST /v2/{project_id}/notifications/sms

A.1.14 MaaS

The SDK interfaces based on the MaaS v1 API are as follows. For details about the invoking methods, see the sample codes.

Interface	method	API URL
Version	osclient.maas().version().get()	GET /v1/{project_id}/objectstorage/version
Task Operations	osclient.maas().task().create(create)	POST /v1/{project_id}/objectstorage/task
	osclient.maas().task().delete(taskId)	DELETE /v1/{project_id}/objectstorage/task/{task_id}
	osclient.maas().task().start(taskId, task)	PUT /v1/{project_id}/objectstorage/task/{task_id}
	osclient.maas().task().stop(taskId)	PUT /v1/{project_id}/objectstorage/task/{task_id}
	osclient.maas().task().count()	GET /v1/{project_id}/objectstorage/task?totalcount=true&state=0
	osclient.maas().task().list(options)	GET /v1/{project_id}/objectstorage/task?start=0&limit=10&state=0
	osclient.maas().task().get(taskId)	GET /v1/{project_id}/objectstorage/task/{task_id}

A.1.15 DMS

The SDK interfaces based on the DMS v1.0 API are as follows. For details about the invoking methods, see the sample codes.

Interface	method	API
Queue	osclient.messageQueue().queue() .create("queue-name", "queue-display-name")	POST /v1.0/{project_id}/queues
	osclient.messageQueue().queue().list()	GET /v1.0/{project_id}/queues
	osclient.messageQueue().queue().get("queue-id")	GET /v1.0/{project_id}/queues/{queue_id}
	osclient.messageQueue().queue().delete("queue-id")	DELETE /v1.0/{project_id}/queues/{queue_id}
Consumer Group	osclient.messageQueue().consumerGroups() .create("queue-id", groupNames)	POST /v1.0/{project_id}/queues/{queue_id}/groups
	osclient.messageQueue().consumerGroups() .list("queue-id")	GET /v1.0/{project_id}/queues/{queue_id}/groups
	osclient.messageQueue().consumerGroups() .delete("queue-id", "consumer-group-id")	DELETE /v1.0/{project_id}/queues/{queue_id}/groups/{consumer_group_id}
Queue Message	osclient.messageQueue().messages() .produce("queue-id", message)	POST /v1.0/{project_id}/queues/{queue_id}/messages
	osclient.messageQueue().messages() .consume("queue-id", "consumer-group-id", maxMessages, timeWait)	GET /v1.0/{project_id}/queues/{queue_id}/groups/{consumer_group_id}/messages
	osclient.messageQueue().messages() .confirmConsuming"queue-id", "consumer-group-id", consumeResult)	POST /v1.0/{project_id}/queues/{queue_id}/groups/{consumer_group_id}/ack
Quota	osclient.messageQueue().quotas().get()	GET /v1.0/{project_id}/quotas/dms

A.2 Python

A.2.1 IAM

The SDK interfaces based on the Keystone v3 API are as follows. Invocation example: conn.identity.endpoints()

Interface	**Method**	**API**

Interface	Method	API
Endpoint Operations	endpoints(self, **query)	GET /v3/endpoints
Project Operations	projects(self, **query)	GET /v3/projects
Service Operations	services(self, **query)	GET /v3/services

A.2.2 IMS

The SDK interfaces based on the Glance v2 API are as follows. Invocation example: conn.image.upload_image()

Interface	Method	API
Image Operations	upload_image(self, container_format=None, disk_format=None, data=None, **attrs)	POST /v2/images PUT /v2/images/{image_id}/file
	delete_image(self, image, ignore_missing=True)	DELETE /v2/images/{image_id}
	find_image(self, name_or_id, ignore_missing=True)	GET /v2/images
	get_image(self, image)	GET /v2/images/{image_id}
	images(self, **query)	GET /v2/images
	add_tag(self, image, tag)	PUT /v2/images/{image_id}/tags/{tag }
	remove_tag(self, image, tag)	DELETE /v2/images/{image_id}/tags/{tag }
Member Operations	add_member(self, image, **attrs)	POST /v2/images/{image_id}/members
	remove_member(self, member, image, ignore_missing=True)	DELETE /v2/images/{image_id}/members /{member_id}
	find_member(self, name_or_id, image, ignore_missing=True)	GET /v2/images/{image_id}/members
	get_member(self, member, image)	GET /v2/images/{image_id}/members /{member_id}
	members(self, image)	GET /v2/images/{image_id}/members
	update_member(self, member, image, **attrs)	PUT /v2/images/{image_id}/members

Interface	Method	API
		/{member_id}

A.2.3 VPC

The SDK interfaces based on the Neutron v2.0 API are as follows.

Invocation example: conn.network.create_network ()

Interface	Method	API
Floating IP Operations	create_ip(self, **attrs)	POST /v2.0/floatingips
	delete_ip(self, floating_ip, ignore_missing=True)	DELETE /v2.0/floatingips/{floatingip_id}
	find_available_ip(self)	GET /v2.0/floatingips
	find_ip(self, name_or_id, ignore_missing=True)	GET /v2.0/floatingips
	get_ip(self, floating_ip)	GET /v2.0/floatingips/{floatingip_id}
	ips(self, **query)	GET /v2.0/floatingips
	update_ip(self, floating_ip, **attrs)	PUT /v2.0/floatingips/{floatingip_id}
Network Operations	create_network(self, **attrs)	POST /v2.0/networks
	delete_network(self, network, ignore_missing=True)	DELETE /v2.0/networks/{network_id}
	find_network(self, name_or_id, ignore_missing=True)	GET /v2.0/networks
	get_network(self, network)	GET /v2.0/networks/{network_id}
	networks(self, **query)	GET /v2.0/networks
	update_network(self, network, **attrs)	PUT /v2.0/networks/{network_id}
Port Operations	create_port(self, **attrs)	POST /v2.0/ports
	delete_port(self, port, ignore_missing=True)	DELETE /v2.0/ports/{port_id}

Interface	Method	API
	find_port(self, name_or_id, ignore_missing=True)	GET /v2.0/ports
	get_port(self, port)	GET /v2.0/ports/{port_id}
	ports(self, **query)	GET /v2.0/ports
	update_port(self, port, **attrs)	PUT /v2.0/ports/{port_id}
Router Operations	create_router(self, **attrs)	POST /v2.0/router
	delete_router(self, router, ignore_missing=True)	DELETE /v2.0/routers/{router_id}
	find_router(self, name_or_id, ignore_missing=True)	GET /v2.0/routers
	get_router(self, router)	GET /v2.0/routers/{router_id}
	routers(self, **query)	GET /v2.0/routers
	update_router(self, router, **attrs)	PUT /v2.0/routers/{router_id}
	add_interface_to_router(self, router, subnet_id=None, port_id=None)	PUT /v2.0/routers/{router_id}/add_router_interface
	remove_interface_from_router(self, router, subnet_id=None, port_id=None)	PUT /v2.0/routers/{router_id}/remove_router_interface
Security Group Operations	create_security_group(self, **attrs)	POST /v2.0/security-groups
	delete_security_group(self, security_group, ignore_missing=True)	DELETE /v2.0/security-groups/{security_group_id}
	find_security_group(self, name_or_id, ignore_missing=True)	GET /v2.0/security-groups
	get_security_group(self, security_group)	GET /v2.0/security-groups/{security_group_id}
	security_groups(self, **query)	GET /v2.0/security-groups
	update_security_group(self, security_group, **attrs)	PUT /v2.0/security-groups/{security_group_id}
	security_group_open_port(self, sgid, port, protocol='tcp')	POST /v2.0/security-group-rules

Interface	Method	API
	security_group_allow_ping(self, sgid)	POST /v2.0/security-group-rules
	create_security_group_rule(self, **attrs)	POST /v2.0/security-group-rules
	delete_security_group_rule(self, security_group_rule, ignore_missing=True)	DELETE /v2.0/security-group-rules/{security_group_rule_id}
	find_security_group_rule(self, name_or_id, ignore_missing=True)	GET /v2.0/security-group-rules
	get_security_group_rule(self, security_group_rule)	GET /v2.0/security-group-rules/{security_group_rule_id}
	security_group_rules(self, **query)	GET /v2.0/security-group-rules
Subnet Operations	create_subnet(self, **attrs)	POST /v2.0/subnets
	delete_subnet(self, subnet, ignore_missing=True)	DELETE /v2.0/subnets/{subnet_id}
	find_subnet(self, name_or_id, ignore_missing=True)	GET /v2.0/subnets
	get_subnet(self, subnet)	GET /v2.0/subnets/{subnet_id}
	subnets(self, **query)	GET /v2.0/subnets
	get_subnet_ports(self, subnet_id)	GET /v2.0/ports
	update_subnet(self, subnet, **attrs)	PUT /v2.0/subnets/{subnet_id}

The SDK interfaces based on the VPC v2.0 API are as follows.

Invocation example: conn.vpc.create_publicip_ext()

Interface	Method	API
Eip Operations	create_publicip_ext(self, **attrs)	POST /v2.0/{project_id}/publicips
Bandwidth Operations	update_bandwidth_ext(self, bandwidth_id, **attrs)	PUT /v2.0/{project_id}/bandwidths/{bandwidth_id}

A.2.4 ECS

The SDK interfaces based on the Nova v2 API are as follows.

Invocation example: conn.compute.create_server()

Interface	Method	API
Flavor Operations	find_flavor(self, name_or_id, ignore_missing=True)	GET /v2/{tenant_id}/flavors
	get_flavor(self, flavor)	GET /v2/{tenant_id}/flavors/{flavor_id}
	flavors(self, details=True, **query)	GET /v2/{tenant_id}/flavors/detail
Image Operations	delete_image(self, image, ignore_missing=True)	DELETE /v2/{tenant_id}/images/{image_id}
	find_image(self, name_or_id, ignore_missing=True)	GET /v2/{tenant_id}/images
	get_image(self, image)	GET /v2/{tenant_id}/images/{image_id}
	images(self, details=True, **query)	GET /v2/{tenant_id}/images/detail
	get_image_metadata(self, image)	GET /v2/{tenant_id}/images/{image_id}/metadata
Keypair Operations	create_keypair(self, **attrs)	POST /v2/{tenant_id}/os-keypairs
	delete_keypair(self, keypair, ignore_missing=True)	DELETE /v2/{tenant_id}/os-keypairs/{keypair_name}
	get_keypair(self, keypair)	GET /v2/{tenant_id}/os-keypairs/{keypair_name}
	find_keypair(self, name_or_id, ignore_missing=True)	GET /v2/{tenant_id}/os-keypairs
	keypairs(self)	GET /v2/{tenant_id}/os-keypairs
Server Operations	create_server(self, **attrs)	POST /v2/{tenant_id}/servers
	delete_server(self, server, ignore_missing=True, force=False)	DELETE /v2/{tenant_id}/servers/{server_id}

Interface	Method	API
	find_server(self, name_or_id, ignore_missing=True)	GET /v2/{tenant_id}/servers
	get_server(self, server)	GET /v2/{tenant_id}/servers/{server_id}
	servers(self, details=True, **query)	GET /v2/{tenant_id}/servers
	update_server(self, server, **attrs)	PUT /v2/{tenant_id}/servers/{server_id}
	reboot_server(self, server, reboot_type)	POST /v2/{tenant_id}/servers/{server_id}/action
	rebuild_server(self, server, name, admin_password, **attrs)	POST /v2/{tenant_id}/servers/{server_id}/action
	resize_server(self, server, flavor)	POST /v2/{tenant_id}/servers/{server_id}/action
	confirm_server_resize(self, server)	POST /v2/{tenant_id}/servers/{server_id}/action
	revert_server_resize(self, server)	POST /v2/{tenant_id}/servers/{server_id}/action
	create_server_image(self, server, name, metadata=None)	POST /v2/{tenant_id}/servers/{server_id}/action
	add_floating_ip_to_server(self, server, address, fixed_address=None)	POST /v2/{tenant_id}/servers/{server_id}/action
	remove_floating_ip_from_server(self, server, address)	POST /v2/{tenant_id}/servers/{server_id}/action
	lock_server(self, server)	POST /v2/{tenant_id}/servers/{server_id}/action
	unlock_server(self, server)	POST /v2/{tenant_id}/servers/{server_id}/action
	start_server(self, server)	POST /v2/{tenant_id}/servers/{server_id}/action

Interface	Method	API
	stop_server(self, server)	POST /v2/{tenant_id}/servers/{server_id }/action
	get_server_metadata(self, server)	GET /v2/{tenant_id}/servers/{server_id }/metadata
	set_server_metadata(self, server, **metadata)	POST /v2/{tenant_id}/servers/{server_id }/metadata
	delete_server_metadata(self, server, keys)	DELETE /v2/{tenant_id}/servers/{server_id }/metadata/{key}
	wait_for_server(self, server, status='ACTIVE', failures=['ERROR'], interval=2, wait=120)	GET /v2/{tenant_id}/servers/{server_id }
Server Interface Operations	create_server_interface(self, server, **attrs)	POST /v2/{tenant_id}/servers/{server_id }/os-interface
	delete_server_interface(self, server_interface, server=None, ignore_missing=True)	DELETE /v2/{tenant_id}/servers/{server_id }/os-interface/{port_id}
	get_server_interface(self, server_interface, server=None)	GET /v2/{tenant_id}/servers/{server_id }/os-interface/{port_id}
	server_interfaces(self, server)	GET /v2/{tenant_id}/servers/{server_id }/os-interface
Server IPs Operations	server_ips(self, server, network_label=None)	GET /v2/{tenant_id}/servers/{server_id }/ips
Availability Zone Operations	availability_zones(self, details=False)	GET /v2/{tenant_id}/os-availability-zon e
Server Group Operations	create_server_group(self, **attrs)	POST /v2/{tenant_id}/os-server-groups
	delete_server_group(self, server_group, ignore_missing=True)	DELETE /v2/{tenant_id}/os-server-groups/{ server_group_id}
	find_server_group(self, name_or_id, ignore_missing=True)	GET /v2/{tenant_id}/os-server-groups
	get_server_group(self, server_group)	GET /v2/{tenant_id}/os-server-groups/{

Interface	Method	API
		server_group_id}

The SDK interfaces based on the ECS v1.1 API are as follows.

Invocation example: conn.ecs.create_server_ext()

Interface	Method	API
Server Operations	create_server_ext(self, **data)	POST /v1.1/{project_id}/cloudservers
	resize_server_ext(self, server_id, **data)	POST /v1.1/{project_id}/cloudservers/{server_id}/resize

A.2.5 EVS

The SDK interfaces based on the Cinder v2 API are as follows.

Invocation example: conn.block_store.create_volume()

Interface	Method	API
Snapshot Operations	get_snapshot(self, snapshot)	GET /v2/{tenant_id}/snapshots/{snapshot_id}
	snapshots(self, details=True, **query)	GET /v2/{tenant_id}/snapshots/detail
	create_snapshot(self, **attrs)	POST /v2/{tenant_id}/snapshots
	delete_snapshot(self, snapshot, ignore_missing=True)	DELETE /v2/{tenant_id}/snapshots/{snapshot_id}
Type Operations	get_type(self, type)	GET /v2/{tenant_id}/types/{volume_type_id}
	types(self)	GET /v2/{tenant_id}/types
Volume Operations	get_volume(self, volume)	GET /v2/{tenant_id}/volumes/{volume_id}
	volumes(self, details=True, **query)	GET /v2/{tenant_id}/volumes/detail
	create_volume(self, **attrs)	POST /v2/{tenant_id}/volumes

Interface	Method	API
	delete_volume(self, volume, ignore_missing=True)	DELETE /v2/{tenant_id}/volumes/{volume_id}

The SDK interfaces based on the EVS v2.1 API are as follows.

Invocation example: conn.evs.create_volume_ext()

Interface	Method	API
Volume Operations	create_volume_ext(self, **attrs)	POST /v2.1/{project_id}/cloudvolumes
	resize_volume_ext(self, volume_id, **data)	POST /v2.1/{project_id}/cloudvolumes/{volume_id}/action

A.2.6 RTS

The SDK interfaces based on the RTS v1 API are as follows. Invocation example: conn.orchestration.create_stack()

Interface	Method	API
Stack Operations	create_stack(self, preview=False, **attrs)	POST /v1/{tenant_id}/stacks
	find_stack(self, name_or_id, ignore_missing=True)	GET /v1/{tenant_id}/stacks
	stacks(self, **query)	GET /v1/{tenant_id}/stacks
	get_stack(self, stack)	GET /v1/{tenant_id}/stacks/{stack_name}/{stack_id}
	update_stack(self, stack, **attrs)	PUT /v1/{tenant_id}/stacks/{stack_name}/{stack_id}
	delete_stack(self, stack, ignore_missing=True)	DELETE /v1/{tenant_id}/stacks/{stack_id}
	check_stack(self, stack)	POST /v1/{tenant_id}/stacks/{stack_name}/{stack_id}/actions
	resources(self, stack, **query)	GET /v1/{tenant_id}/stacks/{stack_name}/{stack_id}/resources

Interface	Method	API
Software_config Operations	create_software_config(self, **attrs)	POST /v1/{tenant_id}/software_configs
	get_software_config(self, software_config)	GET /v1/{tenant_id}/software_configs/{config_id}
	delete_software_config(self, software_config, ignore_missing=True)	DELETE /v1/{tenant_id}/software_configs/{config_id}

A.2.7 AS

The SDK interfaces based on the AS v1 API are as follows. Invocation example: conn.auto_scaling.create_group()

Interface	Method	API
Group Operations	create_group(self, **attrs)	POST /autoscaling-api/v1/{tenant_id}/scaling_group
	groups(self, **query)	GET /autoscaling-api/v1/{tenant_id}/scaling_group
	get_group(self, group)	GET /autoscaling-api/v1/{tenant_id}/scaling_group/{scaling_group_id}
	update_group(self, group, **attrs)	PUT /autoscaling-api/v1/{tenant_id}/scaling_group/{scaling_group_id}
	delete_group(self, group, ignore_missing=True)	DELETE /autoscaling-api/v1/{tenant_id}/scaling_group/{scaling_group_id}
	resume_group(self, group)	POST /autoscaling-api/v1/{tenant_id}/scaling_group/{scaling_group_id}/action
	pause_group(self, group)	POST /autoscaling-api/v1/{tenant_id}/scaling_group/{scaling_group_id}/action
Config Operations	create_config(self, name, **attrs)	POST /autoscaling-api/v1/{tenant_id}/scaling_configuration
	configs(self, **query)	GET /autoscaling-api/v1/{tenant_id}/sc

Interface	Method	API
		aling_configuration
	get_config(self, config)	GET /autoscaling-api/v1/{tenant_id}/scaling_configuration/{scaling_configuration_id}
	delete_config(self, config, ignore_missing=True)	DELETE /autoscaling-api/v1/{tenant_id}/scaling_configuration/{scaling_configuration_id}
	batch_delete_configs(self, configs)	POST /autoscaling-api/v1/{tenant_id}/scaling_configurations
Instance Operations	instances(self, group, **query)	GET /autoscaling-api/v1/{tenant_id}/scaling_group_instance/{scaling_group_id}/list
	remove_instance(self, instance, delete_instance=False, ignore_missing=True)	DELETE /autoscaling-api/v1/{tenant_id}/scaling_group_instance/{instance_id}
	batch_add_instances(self, group, instances)	POST /autoscaling-api/v1/{tenant_id}/scaling_group_instance/{scaling_group_id}/action
	batch_remove_instances(self, group, instances, delete_instance=False)	POST /autoscaling-api/v1/{tenant_id}/scaling_group_instance/{scaling_group_id}/action
Policy Operations	create_policy(self, **attrs)	POST /autoscaling-api/v1/{tenant_id}/scaling_policy
	update_policy(self, policy, **attrs)	PUT /autoscaling-api/v1/{tenant_id}/scaling_policy/{scaling_policy_id}
	policies(self, group, **query)	GET /autoscaling-api/v1/{tenant_id}/scaling_policy/{scaling_group_id}/list
	get_policy(self, policy)	GET /autoscaling-api/v1/{tenant_id}/scaling_policy/{scaling_policy_id}
	execute_policy(self, policy)	POST /autoscaling-api/v1/{tenant_id}/scaling_policy/{scaling_policy_id}/

Interface	Method	API
		action
	resume_policy(self, policy)	POST /autoscaling-api/v1/{tenant_id}/scaling_policy/{scaling_policy_id}/action
	pause_policy(self, policy)	POST /autoscaling-api/v1/{tenant_id}/scaling_policy/{scaling_policy_id}/action
	delete_policy(self, policy, ignore_missing=True)	DELETE /autoscaling-api/v1/{tenant_id}/scaling_policy/{scaling_policy_id}
Activity Operations	activities(self, group, **query)	GET /autoscaling-api/v1/{tenant_id}/scaling_activity_log/{scaling_group_id}
Quota Operations	quotas(self, group=None)	GET /autoscaling-api/v1/{tenant_id}/quotas
	quotas(self, group=None)	GET /autoscaling-api/v1/{tenant_id}/quotas/{scaling_group_id}
Lifecycle_hook Operations	create_lifecycle_hook(self, group, **attrs)	POST /autoscaling-api/v1/{tenant_id}/scaling_lifecycle_hook/{scaling_group_id}
	lifecycle_hooks(self, group)	GET /autoscaling-api/v1/{tenant_id}/scaling_lifecycle_hook/{scaling_group_id}/list
	get_lifecycle_hook(self, group, lifecycle_hook)	GET /autoscaling-api/v1/{tenant_id}/scaling_lifecycle_hook/{scaling_group_id}/{lifecycle_hook_name}
	update_lifecycle_hook(self, group, lifecycle_hook, **attrs)	PUT /autoscaling-api/v1/{tenant_id}/scaling_lifecycle_hook/{scaling_group_id}/{lifecycle_hook_name}
	delete_lifecycle_hook(self, group, lifecycle_hook)	DELETE /autoscaling-api/v1/{tenant_id}/scaling_lifecycle_hook/{scaling_group_id}/{lifecycle_hook_name}
	call_back_instance(self, group, **attrs)	PUT /autoscaling-api/v1/{tenant_id}/scaling_instance_hook/{scaling_gro

Interface	Method	API
		up_id}/callback
	get_group_hanging_instance(self, group, **query)	GET /autoscaling-api/v1/{tenant_id}/scaling_instance_hook/{scaling_group_id}/list{?instance_id}

A.2.8 CES

The SDK interfaces based on the CES v1.0 API are as follows. Invocation example: conn.cloud_eye.metrics()

Interface	Method	API
Metric Operations	metrics(self, **query)	GET /V1.0/{project_id}/metrics
Alarm Operations	alarms(self, **query)	GET /V1.0/{project_id}/alarms
	get_alarm(self, alarm)	GET /V1.0/{project_id}/alarms/{alarm_id}

	enable_alarm(self, alarm)
	disable_alarm(self, alarm)
		PUT /V1.0/{project_id}/alarms/{alarm_id}/action

	delete_alarm(self, alarm, ignore_missing=True)	DELETE /V1.0/{project_id}/alarms/{alarm_id}
Metric Data Operations	metric_aggregations(self, **query)	GET /V1.0/{project_id}/metric-data
	add_metric_data(self, data)	POST /V1.0/{project_id}/metric-data
Quota Operations	quotas(self)	GET /V1.0/{project_id}/quotas

A.2.9 DNS

The SDK interfaces based on the DNS v2 API are as follows. Invocation example: conn.dns.create_zone()

Interface	Method	API
Zone	create_zone(self, **attrs)	POST /v2/zones

Interface	Method	API
Operations	get_zone(self, zone)	GET /v2/zones/{zone_id}
	zones(self, **query)	GET /v2/zones
	delete_zone(self, zone, ignore_missing=True)	DELETE /v2/zones/{zone_id}
	nameservers(self, zone)	GET /v2/zones/{zone_id}/nameservers
	add_router_to_zone(self, zone, **router)	POST /v2/zones/{zone_id}/associaterouter
	remove_router_from_zone(self, zone, **router)	POST /v2/zones/{zone_id}/disassociaterouter
Recordset Operations	create_recordset(self, zone, **attrs)	POST /v2/zones/{zone_id}/recordsets
	get_recordset(self, zone, recordset)	GET /v2/zone/{zone_id}/recordsets/{recordset_id}
	all_recordsets(self, **query)	GET /v2/recordsets
	recordsets(self, zone, **query)	GET /v2/zones/{zone_id}/recordsets
	delete_recordset(self, zone, recordset, ignore_missing=True)	DELETE /v2/zones/{zone_id}/recordsets/{recordset_id}
PTR Record Operations	create_ptr(self, **attrs)	PATCH /v2/reverse/floatingips/{region}:{floatingip_id}
	restore_ptr(self, region, floating_ip_id)	PATCH /v2/reverse/floatingips/{region}:{floatingip_id}
	ptrs(self, **query)	GET /v2/reverse/floatingips
	get_ptr(self, region, floating_ip_id)	GET /v2/reverse/floatingips/{region}:{floatingip_id}

A.2.10 ELB

The SDK interfaces based on the ELB v1.0 API are as follows. Invocation example: conn.load_balancer.create_load_balancer()

Interface	Method	API
LoadBalancer Operations	create_load_balancer(self, **attrs)	POST /v1.0/{tenant_id}/elbaas/loadbalancers
	get_load_balancer(self, load_balancer)	GET /v1.0/{tenant_id}/elbaas/loadbalancers/{loadbalancer_id}
	load_balancers(self, **query)	GET /v1.0/{tenant_id}/elbaas/loadbalancers
	update_load_balancer(self, load_balancer, **attrs)	PUT /v1.0/{tenant_id}/elbaas/loadbalancers/{loadbalancer_id}
	delete_load_balancer(self, load_balancer, ignore_missing=True)	DELETE /v1.0/{tenant_id}/elbaas/loadbalancers/{loadbalancer_id}
Listener Operations	create_listener(self, **attrs)	POST /v1.0/{tenant_id}/elbaas/listeners
	get_listener(self, listener)	GET /v1.0/{tenant_id}/elbaas/listeners/{listener_id}
	listeners(self, **query)	GET /v1.0/{tenant_id}/elbaas/listeners?loadbalancer_id={loadbalancer_id}
	update_listener(self, listener, **attrs)	PUT /v1.0/{tenant_id}/elbaas/listeners/{listener_id}
	delete_listener(self, listener, ignore_missing=True)	DELETE /v1.0/{tenant_id}/elbaas/listeners/{listener_id}
HealthCheck Operations	create_health_check(self, **attrs)	POST /v1.0/{tenant_id}/elbaas/healthcheck
	get_health_check(self, health_check)	GET /v1.0/{tenant_id}/elbaas/healthcheck/{healthcheck_id}
	update_health_check(self, health_check, **attrs)	PUT /v1.0/{tenant_id}/elbaas/healthcheck/{healthcheck_id}
	delete_health_check(self, health_check, ignore_missing=True)	DELETE /v1.0/{tenant_id}/elbaas/healthcheck/{healthcheck_id}

Interface	Method	API
Member Operations	add_members_to_listener(self, listener, members)	POST /v1.0/{tenant_id}/elbaas/listeners/{listener_id}/members
	remove_members_of_listener(self, listener, members)	POST /v1.0/{tenant_id}/elbaas/listeners/{listener_id}/members/action
	listener_members(self, listener, **query)	GET /v1.0/{tenant_id}/elbaas/listeners/{listener_id}/members
Certificate Operations	create_certificate(self, **attrs)	POST /v1.0/{tenant_id}/elbaas/certificate
	certificates(self)	GET /v1.0/{tenant_id}/elbaas/certificate
	update_certificate(self, certificate, **attrs)	PUT /v1.0/{tenant_id}/elbaas/certificate/{certificate_id}
	delete_certificate(self, certificate, ignore_missing=True)	DELETE /v1.0/{tenant_id}/elbaas/certificate/{certificate_id}

A.2.11 VBS

The SDK interfaces based on the VBS v2 API are as follows. Invocation example: conn.volume_backup.create_backup()

Interface	Method	API
VolumeBackup Operations	create_backup(**backup)	POST /v2/{tenant_id}/cloudbackups
	create_native_backup(**backup)	Post /v2/{project_id}/backups
	restore_backup(volume_backup_id, volume_id)	POST/v2/{tenant_id}/cloudbackups/{backup_id}/restore
	backups(**query)	GET /v2/{tenant_id}/backups
	backups(details=True, **query)	GET /v2/{tenant_id}/backups/detail
	get_backup(volume_backup_id)	GET /v2/{tenant_id}/backups/{backup_id}
	delete_backup("volume_backup	DELETE

Interface	Method	API
	_id")	/v2/{tenant_id}/backups/{backup_id}
	get_job("job_id")	GET /v1/{tenant_id}/jobs/{job_id}
VolumeBackupPolicy Operations	create_backup_policy(volume_backup_name, **data)	POST /v2/{tenant_id}/backuppolicy
	backup_policies()	GET /v2/{tenant_id}/backuppolicy
	update_backup_policy(policy, **updated)	PUT /v2/{tenant_id}/backuppolicy/{policy_id}
	delete_backup_policy(policy)	DELETE /v2/{tenant_id}/backuppolicy/{policy_id}
	link_resources_to_policy(policy, volumes)	POST /v2/{tenant_id}/backuppolicyresources
	unlink_resources_of_policy(policy, volumes)	POST /v2/{tenant_id}/backuppolicyresources/{policy_id}/deleted_resources
	execute_policy(policy)	POST /v2/{tenant_id}/backuppolicy/{policy_id}/action
	tasks(backup_policy_id, **query)	GET /v2/{tenant_id}/backuppolicy/{policy_id}/backuptasks
	enable_policy(policy)	PUT /v2/{tenant_id}/backuppolicy/{policy_id}
	disable_policy(policy)	PUT /v2/{tenant_id}/backuppolicy/{policy_id}

A.2.12 CTS

The SDK interfaces based on the CTS v1.0 API are as follows. Invocation example: conn.cts.create_tracker()

Interface	method	API URL
Tracker Operations	create_tracker(**kwargs)	POST /v1.0/{project_id}/tracker

Interface	method	API URL
	get_tracker(name='system')	GET/v1.0/{project_id}/tracker{?tracker_name}
	update_tracker(tracker='system',**kwargs)	PUT/v1.0/{project_id}/tracker/{?tracker_name}
	delete_tracker(tracker='system',ignore_missing=True)	DELETE/v1.0/{project_id}/tracker{?tracker_name}
Trace Operations	traces_v2(tracker='system',**query)	GET/v2.0/{project_id}/{tracker_name}/trace{?trace_id,service_type,resource_type,resource_id,resource_name,trace_name,trace_rating,user,limit,from,to,next}

A.2.13 SMN

The SDK interfaces based on the SMN v2 API are as follows. Invocation example: conn.smn.create_topic()

Interface	method	API URL
Topic Operations	create_topic(**kwargs)	POST /v2/{project_id}/notifications/topics
	update_topic(topic, **kwargs)	PUT /v2/{project_id}/notifications/topics/{topic_urn}
	delete_topic(topic, ignore_missing=True)	DELETE /v2/{project_id}/notifications/topics/{topic_urn}
	topics(**query)	GET /v2/{project_id}/notifications/topics?offset=0&limit=2
	get_topic(topic)	GET /v2/{project_id}/notifications/topics/{topic_urn}
	get_topic_attr(topic, attrname=None)	GET /v2/{project_id}/notifications/topics/{topic_urn}/attributes?name=access_policy
	update_topic_attr(topic_attr, attrname, value)	PUT /v2/{project_id}/notifications/topics/{topic_urn}/attributes/{attributes_name}

Interface	method	API URL
	delete_topic_attr(topic_attr, attrname)	DELETE /v2/{project_id}/notifications/topics/{topic_urn}/attributes/{attributes_name}
	delete_topic_attrs(topic)	DELETE /v2/{project_id}/notifications/topics/{topic_urn}/attributes
Subscribe Operations	subscriptions(**query)	GET /v2/{project_id}/notifications/subscriptions?offset=0&limit=2
	topic_subscriptions(topic, **query)	GET /v2/{project_id}/notifications/topics/{topic_urn}/subscriptions?offset=0&limit=10
	subscript_topic(topic, **kwargs)	POST /v2/{project_id}/notifications/topics/{topic_urn}/subscriptions
	unsubscript_topic(sub, ignore_missing=True)	DELETE /v2/{project_id}/notifications/subscriptions/{subscription_urn}
Message template Operations	create_message_template(**kwargs)	POST /v2/{project_id}/notifications/message_template
	update_message_template(getm, **kwargs)	PUT /v2/{project_id}/notifications/message_template/{message_template_Id}
	delete_message_template(mt,ignore_missing=True)	DELETE /v2/{project_id}/notifications/message_template/{message_template_id}
	message_templates(**query)	GET /v2/{project_id}/notifications/message_template
	get_message_template(mt)	GET /v2/{project_id}/notifications/message_template/{message_template_id}
Publish message Operations	publish_topic(topic, **message_dict)	POST /v2/{project_id}/notifications/topics/{topic_urn}/publish

Interface	method	API URL
Sms Operations	send_sms(conn)	POST /v2/{project_id}/notifications/sms

A.2.14 MaaS

The SDK interfaces based on the MaaS v1 API are as follows. Invocation example: conn.maas.versions()

Interface	method	API URL
Version Operations	versions()	GET /v1/{project_id}/objectstorage/version
Task Operations	create_task(**kwargs)	POST /v1/{project_id}/objectstorage/task
	delete_task(task,ignore_missing=True)	DELETE /v1/{project_id}/objectstorage/task/{task_id}
	start_task(task, source_ak, source_sk, target_ak, target_sk)	PUT /v1/{project_id}/objectstorage/task/{task_id}
	stop_task(task)	PUT /v1/{project_id}/objectstorage/task/{task_id}
	task_count()	GET /v1/{project_id}/objectstorage/task?totalcount=true&state=0
	tasks(**query)	GET /v1/{project_id}/objectstorage/task?start=0&limit=10&state=0
	get_task(task_id)	GET /v1/{project_id}/objectstorage/task/{task_id}

A.2.15 DMS

The SDK interfaces based on the DMS v1.0 API are as follows. Invocation example: conn.dms.create_queue()

Interface	method	API

Queue Operations	create_queue(**kwargs)	POST /v1.0/{project_id}/queues
	queues()	GET /v1.0/{project_id}/queues
	get_queue(queue)	GET /v1.0/{project_id}/queues/{queue_id}
	delete_queue(queue, ignore_missing=True)	DELETE /v1.0/{project_id}/queues/{queue_id}
Group Operations	create_groups(queue, **kwargs)	POST /v1.0/{project_id}/queues/{queue_id}/groups
	groups(queue)	GET /v1.0/{project_id}/queues/{queue_id}/groups
	delete_group(queue, group)	DELETE /v1.0/{project_id}/queues/{queue_id}/groups /{consumer_group_id}
Message Operations	send_messages(queue, **kwargs)	POST /v1.0/{project_id}/queues/{queue_id}/messag es
	consume_message(queue, consume_group, **query)	GET /v1.0/{project_id}/queues/{queue_id}/groups /{consumer_group_id}/messages
	ack_consumed_message(c onsumed_message, status='success')	POST /v1.0/{project_id}/queues/{queue_id}/groups /{consumer_group_id}/ack
Quota Operations	quotas()	GET /v1.0/{project_id}/quotas/dms

B OpenStack Client CLI Command

For command details, see **OpenStack Client CLI Command** in API List.

C Change History

Released On	Description
2018-07-30	This issue is the third official release, which incorporates the following change: Added the SDKs of RTS, CTS, SMN, MaaS, and DMS.
2018-04-30	This issue is the second official release, which incorporates the following change: Added SDKs.
2017-07-29	This issue is the first official release.