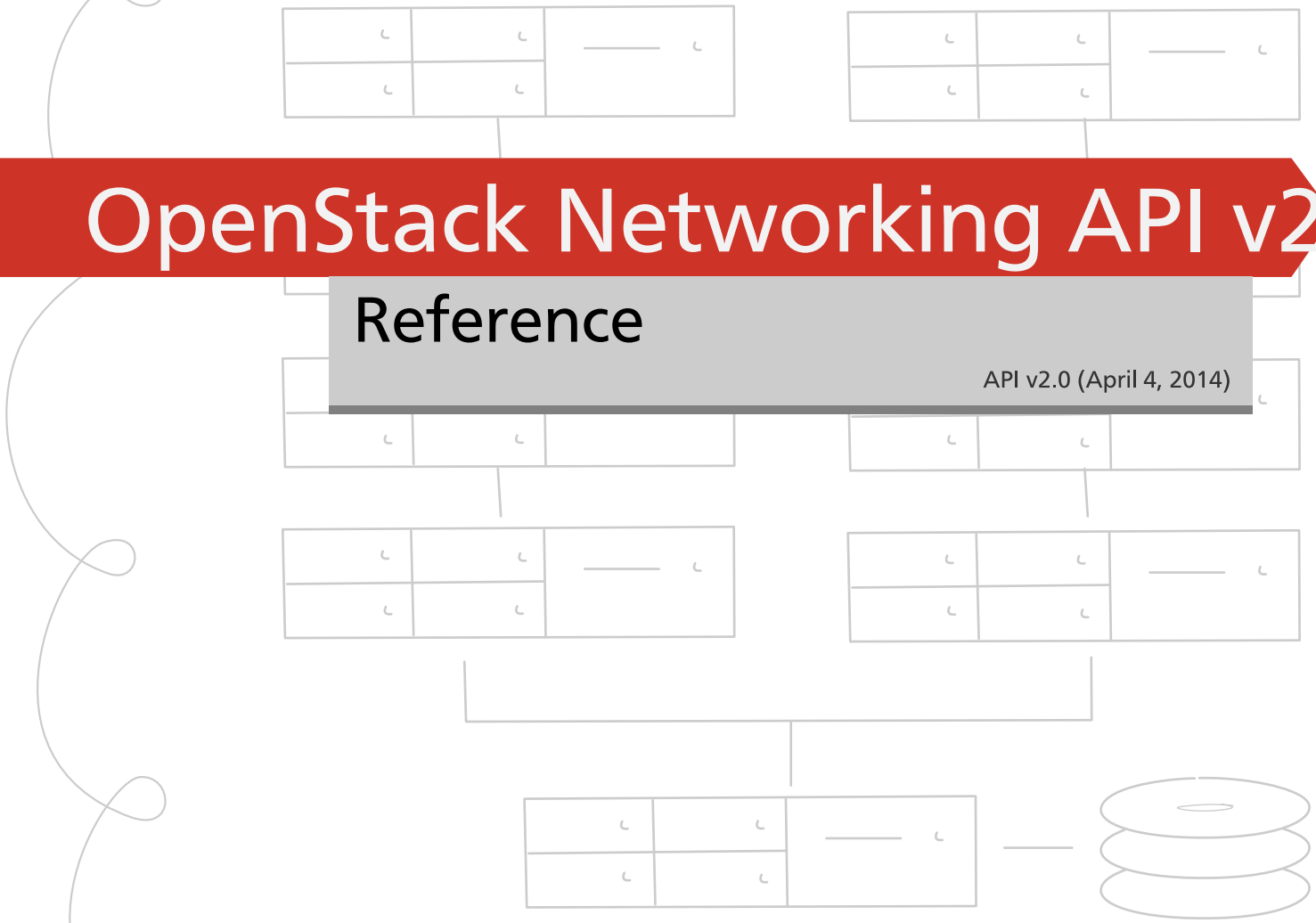
OpenStack Networking API v2.0

Reference

API v2.0 (April 4, 2014)



BUILT FOR

openstack™

CLOUD SOFTWARE

openstack™

docs.openstack.org

OpenStack Networking API v2.0 Reference

API v2.0 (2014-04-04)

This document is for software developers who develop applications by using the OpenStack Networking API v2.0.

Table of Contents

Preface

The OpenStack Networking project provides virtual networking services among devices managed by the OpenStack compute service.

This document describes the Networking API v2.0 features.

We welcome feedback, comments, and bug reports at bugs.launchpad.net/Neutron.

Intended Audience

This guide is for software developers who create applications by using the Networking API v2.0. To use this information, you should have a general understanding of the OpenStack Networking service, the OpenStack compute service, and the integration between the two. You should also have access to a plug-in that implements the Networking API v2.0.

You should also be familiar with:

- ReSTful web services
- HTTP/1.1
- JSON and XML data serialization formats

Document Change History

This version of the document replaces and obsoletes all previous versions. The following table describes the most recent changes:

Revision Date	Summary of Changes
December 20, 2013	• Updated book to source information from WADL files.
October 11, 2013	• Added the Networking API ports binding extension.
May 22, 2013	• Updated the title of the book and project name to "OpenStack Networking." • Updated the title of the book to *Reference* from *Developer Guide* for consistency.
March 23, 2013	• Updated incorrect nova command examples in the section called “High-Level Task Flow” [10].
September 5, 2012	• Removed XML as a valid request or response type.
August 17, 2012	• First edition of this book.

Resources

Use the following resources in conjunction with this guide:

Resource	See
Related documents	OpenStack Manuals
OpenStack Neutron Wiki	http://wiki.openstack.org/Neutron

1. Overview

The Neutron project provides virtual networking services among devices that are managed by the OpenStack compute service.

The Networking API v2.0 combines the Quantum API v1.1 with some essential Internet Protocol Address Management (IPAM) capabilities from the Melange API.

These IPAM capabilities enable you to:

- Associate IP address blocks and other network configuration settings required by a network device, such as a default gateway and dns-servers settings, with an OpenStack Networking network.
- Allocate an IP address from a block and associate it with a device that is attached to the network through an OpenStack Networking port.

To do this, the Networking API v2.0 introduces the subnet entity. A subnet can represent either an IP version 4 or version 6 address block. Each OpenStack Networking network commonly has one or more subnets. When you create a port on the network, an available fixed IP address is allocated to it from one the designated subnets for each IP version. When you delete the port, the allocated addresses return to the pool of available IPs on the subnet. Networking API v2.0 users can choose a specific IP address from the block or let OpenStack Networking choose the first available IP address.

Note

The Quantum API v1.x was removed from the source code tree. To use the Quantum API v1.x, install the Quantum Essex release.

Glossary

Term	Description
CRUD	Create, read, update, and delete (CRUD) are the basic functions of persistent storage in computer programming.
entity	Any piece of hardware or software that can connect to the network services provided by OpenStack Networking. An entity can use OpenStack Networking services by implementing a VIF.
IPAM	Internet Protocol Address Management (IPAM) is a means of planning, tracking, and managing the Internet Protocol (IP) address space that is used in a network.
layer-2 network	A virtual Ethernet network that is managed by the OpenStack Networking service. Currently, OpenStack Networking manages only Ethernet networks.
network	An isolated virtual layer-2 broadcast domain that is typically reserved for the tenant who created it unless the network is configured to be shared. Tenants can create multiple networks until they reach the thresholds specified by per-tenant quotas.
plug-in	A software component that implements Networking API v2.0.
port	A virtual switch port on a logical network switch. Virtual instances attach their interfaces into ports. The logical port also defines the MAC address and the IP addresses to be assigned to the interfaces plugged into them. When IP addresses are associated to a port, this also implies the port is associated with a subnet, as the IP address was taken from the allocation pool for a specific subnet.
subnet	An IP address block that can be used to assign IP addresses to virtual instances. Each subnet must have a CIDR and must be associated with a network. IPs can be either selected from the whole subnet CIDR or from allocation pools that can be specified by the user.
VM	A virtual machine (VM) is a completely isolated guest operating system installation within a normal host operating system.

Concepts

Use the Networking API v2.0 to manage the following entities:

- **Network**. An isolated virtual layer-2 domain. A network can also be a virtual, or logical, switch. See the section called "Network" [4].
- **Subnet**. An IP version 4 or version 6 address block from which IP addresses that are assigned to VMs on a specified network are selected. See the section called "Subnet" [6].
- **Port**. A virtual, or logical, switch port on a specified network. See the section called "Port" [8].

These entities have auto-generated unique identifiers and support basic create, read, update, and delete (CRUD) functions with the **POST**, **GET**, **PUT**, and **DELETE** verbs.

Network

A network is an isolated virtual layer-2 broadcast domain that is typically reserved for the tenant who created it unless you configure the network to be shared. Tenants can create multiple networks until the thresholds per-tenant quota is reached.

In the Networking API v2.0, the network is the main entity. Ports and subnets are always associated with a network.

The following table describes the attributes for network objects:

Table 1.1. Network Attributes

Attribute	Type	Required	CRUD[a]	Default Value	Validation Constraints	Notes
id	uuid-str	N/A	R	generated	N/A	UUID for the network.
name	String	No	CRU	None	N/A	Human-readable name for the network. Might not be unique.
admin_state_up	Bool	No	CRU	true	{true|false}	The administrative state of network. If false (down), the network does not forward packets.
status	String	N/A	R	N/A	N/A	Indicates whether network is currently operational. Possible values include: • ACTIVE • DOWN • BUILD • ERROR Plug-ins might define additional values.
subnets	list(uuid-str)	No	R	Empty List	N/A	subnets associated with this network.
shared	Bool	No	CRU	False	{ True | False }	Specifies whether the network resource can be accessed by any tenant or not.
tenant_id	uuid-str	No[b]	CR	N/A	No constraint	Owner of network. Only admin users can specify a tenant_id other than its own.

[a]

- **C**. Use the attribute in create operations.
- **R**. This attribute is returned in response to show and list operations.
- **U**. You can update the value of this attribute.
- **D**. You can delete the value of this attribute.

[b] If OpenStack Networking is not running with the Keystone Identity service, the `tenant_id` attribute is required.

Subnet

A subnet represents an IP address block that can be used to assign IP addresses to virtual instances. Each subnet must have a CIDR and must be associated with a network. IPs can be either selected from the whole subnet CIDR or from allocation pools that can be specified by the user.

A subnet can also optionally have a gateway, a list of dns name servers, and host routes. This information is pushed to instances whose interfaces are associated with the subnet.

Table 1.2. Subnet Attributes

Attribute	Type	Required	CRUD[a]	Default Value	Validation Constraints	Notes
id	uuid-str	N/A	R	generated	N/A	UUID representing the subnet
network_id	uuid-str	Yes	CR	N/A	network this subnet is associated with.	
name	String	No	CRU	None	N/A	Human-readable name for the subnet. Might not be unique.
ip_version	int	Yes	CR	4	{ 4 | 6 }	IP version
cidr	string	Yes	CR	N/A	valid cidr in the form <network_address>/<prefix>	cidr representing IP range for this subnet, based on IP version
gateway_ip	string	No	CRUD	first address in *cidr*	Valid IP address or null	default gateway used by devices in this subnet
dns_nameservers	list(str)	No	CRU	Empty list	No constraint	DNS name servers used by hosts in this subnet.
allocation_pools	list(dict)	No	CR	Every address in *cidr*, excluding *gateway_ip* if configured	star/end of range must be valid ip	Sub-ranges of cidr available for dynamic allocation to ports [{ "start": "10.0.0.2", "end": "10.0.0.254"}]
host_routes	list(dict)	No	CRU	Empty List	*[]*	Routes that should be used by devices with IPs from this subnet (not including local subnet route).
enable_dhcp	Bool	No	CRU	True	{ True | False }	Specifies whether DHCP is enabled for this subnet or not.
tenant_id	uuid-str	No[b]	CR	N/A	No constraint	Owner of network. Only admin users can specify a tenant_id other than its own.

[a]

- **C**. Use the attribute in create operations.
- **R**. This attribute is returned in response to show and list operations.
- **U**. You can update the value of this attribute.
- **D**. You can delete the value of this attribute.

[b] If OpenStack Networking is not running with the Keystone Identity service, the `tenant_id` attribute is required.

Port

A port represents a virtual switch port on a logical network switch. Virtual instances attach their interfaces into ports. The logical port also defines the MAC address and the IP address(es) to be assigned to the interfaces plugged into them. When IP addresses are associated to a port, this also implies the port is associated with a subnet, as the IP address was taken from the allocation pool for a specific subnet.

Table 1.3. Port Attributes

Attribute	Type	Required	CRUD[a]	Default Value	Validation Constraints	Notes
id	uuid-str	N/A	R	generated	N/A	UUID for the port.
network_id	uuid-str	Yes	CR	N/A	existing network identifier	Network that this port is associated with.
name	String	No	CRU	None	N/A	Human-readable name for the port. Might not be unique.
admin_state_up	bool	No	CRU	true	{true|false}	Administrative state of port. If false (down), port does not forward packets.
status	string	N/A	R	N/A	N/A	Indicates whether network is currently operational. Possible values include: • ACTIVE • DOWN • BUILD • ERROR Plug-ins might define additional values.
mac_address	string	No	CR	generated	valid MAC in 6-octet form separated by colons	Mac address to use on this port.
fixed_ips	list(dict)	No	CRU	automatically allocated from pool	Valid IP address and existing subnet identifier	Specifies IP addresses for the port thus associating the port itself with the subnets where the IP addresses are picked from
device_id	str	No	CRUD	None	No constraint	identifies the device (e.g., virtual server) using this port.
device_owner	str	No	CRUD	None	No constraint	Identifies the entity (e.g.: dhcp agent) using this port.
tenant_id	uuid-str	No[b]	CR	N/A	No constraint	Owner of network. Only admin users can specify a tenant_id other than its own.
security_groups	list(dict)	No	CRUD	None	Existing security group IDs	Specifies the IDs of any security groups associated with a port.

[a]

- **C**. Use the attribute in create operations.
- **R**. This attribute is returned in response to show and list operations.
- **U**. You can update the value of this attribute.

- **D**. You can delete the value of this attribute.

[b]If OpenStack Networking is not running with the Keystone Identity service, the `tenant_id` attribute is required.

High-Level Task Flow

The high-level task flow for OpenStack Networking involves creating a network, associating a subnet with that network, and booting a VM that is attached to the network. Clean-up includes deleting the VM, deleting any ports associated with the network, and deleting the networks. OpenStack Networking deletes any subnets associated with the deleted network.

To use OpenStack Networking - high-level task flow

1. **Create a network**

   The tenant creates a network.

   For example, the tenant creates the `net1` network. Its ID is *net1_id*.

2. **Associate a subnet with the network**

   The tenant associates a subnet with that network.

   For example, the tenant associates the `10.0.0.0/24` subnet with the `net1` network.

3. **Boot a VM and attach it to the network**

   The tenant boots a virtual machine (VM) and specifies a single NIC that connects to the network.

   The following examples use the nova client to boot a VM.

   In the first example, Nova contacts OpenStack Networking to create the NIC and attach it to the `net1` network, with the ID *net1_id*:

   ```
   $ nova boot <server_name> --image <image> --flavor <flavor> --nic net-id=
   <net1_id>
   ```

   In a second example, you first create the `port1`, port and then you boot the VM with a specified port. OpenStack Networking creates a NIC and attaches it to the `port1` port, with the ID *port1_id*:

   ```
   $ nova boot <server_name> --image <image> --flavor <flavor> --nic port-id=
   <port1_id>
   ```

   OpenStack Networking chooses and assigns an IP address to the `port1` port.

   For more information about the **nova boot** command, enter:

   ```
   $ nova help boot
   ```

4. **Delete the VM**

   The tenant deletes the VM.

Nova contacts OpenStack Networking and deletes the `port1` port.

The allocated IP address is returned to the pool of available IP addresses.

5. **Delete any ports**

   If the tenant created any ports and associated them with the network, the tenant deletes the ports.

6. **Delete the network**

   The tenant deletes the network. This operation deletes an OpenStack Networking network and its associated subnets provided that no port is currently configured on the network.

Plug-ins

Virtual networking services are implemented through a plug-in. A plug-in can use different techniques and technologies to provide isolated virtual networks to tenants. A plug-in also provides other services, such as IP address management. Because each plug-in implements all the operations included in Networking API v2.0, do not be concerned about which plug-in is used.

However, some plug-ins might expose additional capabilities through API extensions, which this document discusses. For more information about the extensions exposed by a particular plug-in, see the plug-in documentation.

2. General API Information

The Networking API v2.0 is a ReSTful HTTP service that uses all aspects of the HTTP protocol including methods, URIs, media types, response codes, and so on. Providers can use existing features of the protocol including caching, persistent connections, and content compression. For example, providers who employ a caching layer can respond with a 203 code instead of a 200 code when a request is served from the cache. Additionally, providers can offer support for conditional **GET** requests by using ETags, or they may send a redirect in response to a **GET** request. Create clients so that these differences are accounted for.

Authentication and Authorization

The Networking API v2.0 uses the Keystone Identity Service as the default authentication service. When Keystone is enabled, users that submit requests to the OpenStack Networking service must provide an authentication token in **X-Auth-Token** request header. You obtain the token by authenticating to the Keystone endpoint. For more information about Keystone, see the *OpenStack Identity Service API v2.0 Reference*.

When Keystone is enabled, the `tenant_id` attribute is not required in create requests because the tenant ID is derived from the authentication token.

The default authorization settings allow only administrative users to create resources on behalf of a different tenant.

OpenStack Networking uses information received from Keystone to authorize user requests. OpenStack Networking handles the following types of authorization policies:

- **Operation-based policies** specify access criteria for specific operations, possibly with fine-grained control over specific attributes.
- **Resource-based policies** access a specific resource. Permissions might or might not be granted depending on the permissions configured for the resource. Currently available for only the network resource.

The actual authorization policies enforced in OpenStack Networking might vary from deployment to deployment.

Request/Response Types

The Networking API v2.0 supports the JSON data serialization format.

The format for both the request and the response can be specified by using the `Content-Type` header, the `Accept` header or adding the `.json` extension to the request URI.

Example 2.1. JSON Request with Headers

```
POST /v1.0/tenants/tenantX/networks HTTP/1.1
Host 127.0.0.1:9696
Content-Type application/json
Accept application/json
Content-Length 57
```

```
{
 "network":
    {
      "name": "net-name",
      "admin_state_up": true
    }
}
```

Example 2.2. JSON Response with Headers

```
HTTP/1.1 201 Created
Content-Type application/json
Content-Length 204
```

```
{
 "network":
    {
     "status": "ACTIVE",
     "subnets": [],
     "name": "net-name",
     "admin_state_up": true,
     "tenant_id": "388a70781bae4ca895f17b7f6293eb70",
     "shared": false, "id": "2a4017ef-31ff-496a-9294-e96ecc3bc9c9"
    }
}
```

Filtering and Column Selection

The Networking API v2.0 supports filtering based on all top level attributes of a resource. Filters are applicable to all list requests.

For example, the following request returns all networks named `foobar`:

```
GET /v2.0/networks?name=foobar
```

When you specify multiple filters, the Networking API v2.0 returns only objects that meet all filtering criteria. The operation applies an AND condition among the filters.

Note

OpenStack Networking does not offer an OR mechanism for filters.

Alternatively, you can issue a distinct request for each filter and build a response set from the received responses on the client-side.

By default, OpenStack Networking returns all attributes for any show or list call. The Networking API v2.0 has a mechanism to limit the set of attributes returned. For example, return `id`.

You can use the `fields` query parameter to control the attributes returned from the Networking API v2.0.

For example, the following request returns only `id` and `name` for each network:

```
GET /v2.0/networks.json?fields=id&fields=name
```

Synchronous versus Asynchronous Plug-in Behavior

The Networking API v2.0 presents a logical model of network connectivity consisting of networks, ports, and subnets. It is up to the OpenStack Networking plug-in to communicate with the underlying infrastructure to ensure packet forwarding is consistent with the logical model. A plug-in might perform these operations asynchronously.

When an API client modifies the logical model by issuing an HTTP **POST**, **PUT**, or **DELETE** request, the API call might return before the plug-in modifies underlying virtual and physical switching devices. However, an API client is guaranteed that all subsequent API calls properly reflect the changed logical model.

For example, if a client issues an HTTP **PUT** request to set the attachment for a port, there is no guarantee that packets sent by the interface named in the attachment are forwarded immediately when the HTTP call returns. However, it is guaranteed that a subsequent HTTP **GET** request to view the attachment on that port returns the new attachment value.

You can use the `status` attribute with the network and port resources to determine whether the OpenStack Networking plug-in has successfully completed the configuration of the resource.

Bulk Create Operations

The Networking API v2.0 enables you to create several objects of the same type in the same API request. Bulk create operations use exactly the same API syntax as single create operations except that you specify a list of objects rather than a single object in the request body.

Bulk operations are always performed atomically, meaning that either all or none of the objects in the request body are created. If a particular plug-in does not support atomic operations, the Networking API v2.0 emulates the atomic behavior so that users can expect the same behavior regardless of the particular plug-in running in the background.

OpenStack Networking might be deployed without support for bulk operations and when the client attempts a bulk create operation, a 400 Bad Request error is returned.

Pagination

To reduce load on the service, list operations will return a maximum number of items at a time. To navigate the collection, the parameters limit, marker and page_reverse can be set in the URI. For example:

```
?limit=100&marker=1234&page_reverse=False
```

The *marker* parameter is the ID of the last item in the previous list. The *limit* parameter sets the page size. The *page_reverse* parameter sets the page direction. These parameters are optional. If the client requests a limit beyond the maximum limit configured by the deployment, the server returns the maximum limit number of items.

For convenience, list responses contain atom "next" links and "previous" links. The last page in the list requested with 'page_reverse=False' will not contain "next" link, and the last page in the list requested with 'page_reverse=True' will not contain "previous" link. The following examples illustrate two pages with three items. The first page was retrieved through:

```
GET http://127.0.0.1:9696/v2.0/networks.json?limit=2
```

Pagination is an optional feature of OpenStack Networking API, and it might be disabled. If pagination is disabled, the pagination parameters will be ignored and return all the items.

If a particular plug-in does not support pagination operations, and pagination is enabled, the Networking API v2.0 will emulate the pagination behavior so that users can expect the same behavior regardless of the particular plug-in running in the background.

Unfortunately OpenStack Networking does not expose any mechanism to tell user if pagination is supported by particular plug-in or enabled.

Example 2.3. Network Collection, First Page: JSON Request

```
GET /v2.0/networks.json?limit=2 HTTP/1.1
Host: 127.0.0.1:9696
Content-Type: application/json
Accept: application/json
```

Example 2.4. Network Collection, First Page: JSON Response

```
{
    "networks": [
        {
            "admin_state_up": true,
            "id": "396f12f8-521e-4b91-8e21-2e003500433a",
            "name": "net3",
            "provider:network_type": "vlan",
            "provider:physical_network": "physnet1",
            "provider:segmentation_id": 1002,
            "router:external": false,
            "shared": false,
            "status": "ACTIVE",
            "subnets": [],
            "tenant_id": "20bd52ff3e1b40039c312395b04683cf"
        },
        {
```

```
            "admin_state_up": true,
            "id": "71c1e68c-171a-4aa2-aca5-50ea153a3718",
            "name": "net2",
            "provider:network_type": "vlan",
            "provider:physical_network": "physnet1",
            "provider:segmentation_id": 1001,
            "router:external": false,
            "shared": false,
            "status": "ACTIVE",
            "subnets": [],
            "tenant_id": "20bd52ff3e1b40039c312395b04683cf"
        }
    ],
    "networks_links": [
        {
            "href": "http://127.0.0.1:9696/v2.0/networks.json?limit=2&marker=
71c1e68c-171a-4aa2-aca5-50ea153a3718",
            "rel": "next"
        },
        {
            "href": "http://127.0.0.1:9696/v2.0/networks.json?limit=2&marker=
396f12f8-521e-4b91-8e21-2e003500433a&page_reverse=True",
            "rel": "previous"
        }
    ]
}
```

Example 2.5. Network Collection, First Page: XML Request

```
GET /v2.0/networks.xml?limit=2 HTTP/1.1
Host: 127.0.0.1:9696
Content-Type: application/xml
Accept: application/xml
```

Example 2.6. Network Collection, First Page: XML Response

```
<?xml version="1.0" ?>
<networks xmlns="http://openstack.org/neutron/api/v2.0" xmlns:atom="http://
www.w3.org/2005/Atom" xmlns:provider="http://docs.openstack.org/ext/provider/
api/v1.0" xmlns:neutron="http://openstack.org/neutron/api/v2.0" xmlns:router=
"http://docs.openstack.org/ext/neutron/router/api/v1.0" xmlns:xsi="http://www.
w3.org/2001/XMLSchema-instance">
        <network>
                <status>ACTIVE</status>
                <subnets neutron:type="list"/>
                <name>net3</name>
                <provider:physical_network>physnet1</
provider:physical_network>
                <admin_state_up neutron:type="bool">True</admin_state_up>
                <tenant_id>20bd52ff3e1b40039c312395b04683cf</tenant_id>
                <provider:network_type>vlan</provider:network_type>
                <router:external neutron:type="bool">False</router:external>
                <shared neutron:type="bool">False</shared>
                <id>396f12f8-521e-4b91-8e21-2e003500433a</id>
                <provider:segmentation_id neutron:type="long">1002</
provider:segmentation_id>
        </network>
```

```
        <network>
                <status>ACTIVE</status>
                <subnets neutron:type="list"/>
                <name>net2</name>
                <provider:physical_network>physnet1</
provider:physical_network>
                <admin_state_up neutron:type="bool">True</admin_state_up>
                <tenant_id>20bd52ff3e1b40039c312395b04683cf</tenant_id>
                <provider:network_type>vlan</provider:network_type>
                <router:external neutron:type="bool">False</router:external>
                <shared neutron:type="bool">False</shared>
                <id>71c1e68c-171a-4aa2-aca5-50ea153a3718</id>
                <provider:segmentation_id neutron:type="long">1001</
provider:segmentation_id>
        </network>
        <atom:link href="http://127.0.0.1:9696/v2.0/networks.xml?limit=2&
amp;marker=71c1e68c-171a-4aa2-aca5-50ea153a3718" rel="next"/>
        <atom:link href="http://127.0.0.1:9696/v2.0/networks.xml?limit=2&
amp;marker=396f12f8-521e-4b91-8e21-2e003500433a&amp;page_reverse=True" rel=
"previous"/>
</networks>
```

The last page won't show the "next" links

Example 2.7. Network Collection, Last Page: JSON Request

```
GET /v2.0/networks.json?limit=2&marker=71c1e68c-171a-4aa2-aca5-50ea153a3718
 HTTP/1.1
Host: 127.0.0.1:9696
Content-Type: application/json
Accept: application/json
```

Example 2.8. Network Collection, Last Page: JSON Response

```
{
    "networks": [
        {
            "admin_state_up": true,
            "id": "b3680498-03da-4691-896f-ef9ee1d856a7",
            "name": "net1",
            "provider:network_type": "vlan",
            "provider:physical_network": "physnet1",
            "provider:segmentation_id": 1000,
            "router:external": false,
            "shared": false,
            "status": "ACTIVE",
            "subnets": [],
            "tenant_id": "c05140b3dc7c4555afff9fab6b58edc2"
        }
    ],
    "networks_links": [
        {
            "href": "http://127.0.0.1:9696/v2.0/networks.json?limit=2&marker=
b3680498-03da-4691-896f-ef9ee1d856a7&page_reverse=True",
            "rel": "previous"
        }
    ]
```

```
}
```

Example 2.9. Network Collection, Last Page: XML Request

```
GET /v2.0/networks.xml?limit=2&marker=71c1e68c-171a-4aa2-aca5-50ea153a3718
 HTTP/1.1
Host: 127.0.0.1:9696
Content-Type: application/xml
Accept: application/xml
```

Example 2.10. Network Collection, Last Page: XML Response

```
<?xml version="1.0" ?>
<networks xmlns="http://openstack.org/neutron/api/v2.0" xmlns:atom="http://
www.w3.org/2005/Atom" xmlns:provider="http://docs.openstack.org/ext/provider/
api/v1.0" xmlns:neutron="http://openstack.org/neutron/api/v2.0" xmlns:router=
"http://docs.openstack.org/ext/neutron/router/api/v1.0" xmlns:xsi="http://www.
w3.org/2001/XMLSchema-instance">
        <network>
                <status>ACTIVE</status>
                <subnets neutron:type="list"/>
                <name>net1</name>
                <provider:physical_network>physnet1</
provider:physical_network>
                <admin_state_up neutron:type="bool">True</admin_state_up>
                <tenant_id>c05140b3dc7c4555afff9fab6b58edc2</tenant_id>
                <provider:network_type>vlan</provider:network_type>
                <router:external neutron:type="bool">False</router:external>
                <shared neutron:type="bool">False</shared>
                <id>b3680498-03da-4691-896f-ef9ee1d856a7</id>
                <provider:segmentation_id neutron:type="long">1000</
provider:segmentation_id>
        </network>
        <atom:link href="http://127.0.0.1:9696/v2.0/networks.xml?limit=2&
amp;marker=b3680498-03da-4691-896f-ef9ee1d856a7&amp;page_reverse=True" rel=
"previous"/>
</networks>
```

Sorting

The results of list operations can be ordered using the 'sort_key' and 'sort_dir' parameters. Currently sorting doesn't work with extended attributes of resource. The 'sort_key' and 'sort_dir' can be repeated, and the number of 'sort_key' and 'sort_dir' provided must be same. The sort_dir parameter indicates in which direction to sort. Acceptable values are 'asc' (ascending) and 'desc' (descending).

Sorting is optional feature of OpenStack Networking API, and it might be disabled. If sorting is disabled, the sorting parameters will be ignored.

If a particular plug-in does not support sorting operations, and sorting is enabled, the Networking API v2.0 will emulate the sorting behavior so that users can expect the same behavior regardless of the particular plug-in running in the background.

Unfortunately OpenStack Networking does provide a mechanism to tell users if specific plug-ins support or have enabled sorting.

Extensions

The Networking API v2.0 is extensible.

The purpose of Networking API v2.0 extensions is to:

- Introduce new features in the API without requiring a version change.
- Introduce vendor-specific niche functionality.
- Act as a proving ground for experimental functionalities that might be included in a future version of the API.

To programmatically determine which extensions are available, issue a **GET** request on the **v2.0/extensions** URI.

To query extensions individually by unique alias, issue a **GET** request on the **/v2.0/extensions/*alias_name*** URI. Use this method to easily determine if an extension is available. If the extension is not available, a 404 Not Found response is returned.

You can extend existing core API resources with new actions or extra attributes. Also, you can add new resources as extensions. Extensions usually have tags that prevent conflicts with other extensions that define attributes or resources with the same names, and with core resources and attributes. Because an extension might not be supported by all plug-ins, the availability of an extension varies with deployments and the specific plug-in in use.

For more information regarding specific extensions, see Chapter 4, "API Extensions" [66]

Faults

The Networking API v2.0 returns an error response if a failure occurs while processing a request. OpenStack Networking uses only standard HTTP error codes. 4xx errors indicate problems in the particular request being sent from the client.

Error	Description	
400	Bad Request	Malformed request URI or body
		Requested admin state invalid
		Invalid values entered
		Bulk operations disallowed
		Validation failed
		Method not allowed for request body (such as trying to update attributes that can be specified at create-time only)
404	Not Found	Non existent URI
		Resource not found
409	Conflict	Port configured on network
		IP allocated on subnet
		Conflicting IP allocation pools for subnet
500	Internal server error	Internal OpenStack Networking error
503	Service unavailable	Failure in Mac address generation

Users submitting requests to the Networking API v2.0 might also receive the following errors:

- 401 Unauthorized - If invalid credentials are provided.
- 403 Forbidden - If the user cannot access a specific resource or perform the requested operation.

3. API Operations

Provides virtual networking services among devices that are managed by the OpenStack Compute service. The Networking API v2.0 combines the API v1.1 functionality with some essential Internet Protocol Address Management (IPAM) functionality.

Enables users to associate IP address blocks and other network configuration settings with a neutron network. You can choose a specific IP address from the block or let neutron choose the first available IP address.

Method	URI	Description
Networks		
GET	`/v2.0/networks`	Lists networks to which the specified tenant has access.
POST	`/v2.0/networks`	Creates a network.
POST	`/v2.0/networks`	Creates multiple networks in a single request.
GET	`/v2.0/networks/{network_id}`	Shows information for a specified network.
PUT	`/v2.0/networks/{network_id}`	Updates a specified network.
DELETE	`/v2.0/networks/{network_id}`	Deletes a specified network and its associated resources.
Subnets		
GET	`/v2.0/subnets`	Lists subnets to which the specified tenant has access.
POST	`/v2.0/subnets`	Creates a subnet on a specified network.
POST	`/v2.0/subnets`	Creates multiple subnets in a single request. Specify a list of subnets in the request body.
GET	`/v2.0/subnets/{subnet_id}`	Shows information for a specified subnet.
PUT	`/v2.0/subnets/{subnet_id}`	Updates a specified subnet.
DELETE	`/v2.0/subnets/{subnet_id}`	Deletes a specified subnet.
Ports		
GET	`/v2.0/ports`	Lists ports to which the tenant has access.
POST	`/v2.0/ports`	Creates a port on a specified network.
POST	`/v2.0/ports`	Creates multiple ports in a single request. Specify a list of ports in the request body.
GET	`/v2.0/ports/{port_id}`	Shows information for a specified port.
PUT	`/v2.0/ports/{port_id}`	Updates a specified port.
DELETE	`/v2.0/ports/{port_id}`	Deletes a specified port.

Networks

List, show information for, create, update, and delete networks.

Method	URI	Description
GET	`/v2.0/networks`	Lists networks to which the specified tenant has access.
POST	`/v2.0/networks`	Creates a network.
POST	`/v2.0/networks`	Creates multiple networks in a single request.

Method	URI	Description
GET	`/v2.0/networks/{network_id}`	Shows information for a specified network.
PUT	`/v2.0/networks/{network_id}`	Updates a specified network.
DELETE	`/v2.0/networks/{network_id}`	Deletes a specified network and its associated resources.

List networks

Method	URI	Description
GET	/v2.0/networks	Lists networks to which the specified tenant has access.

You can control which attributes are returned by using the fields query parameter. For information, see Filtering and Column Selection in the *OpenStack Networking API v2.0 Reference*.

Normal response codes: 200

Error response codes: unauthorized (401)

Request

This operation does not require a request body.

Response

Example 3.1. List networks: JSON response

```
{
   "networks":[
      {
         "status":"ACTIVE",
         "subnets":[
            "54d6f61d-db07-451c-9ab3-b9609b6b6f0b"
         ],
         "name":"private-network",
         "provider:physical_network":null,
         "admin_state_up":true,
         "tenant_id":"4fd44f30292945e481c7b8a0c8908869",
         "provider:network_type":"local",
         "router:external":true,
         "shared":true,
         "id":"d32019d3-bc6e-4319-9c1d-6722fc136a22",
         "provider:segmentation_id":null
      },
      {
         "status":"ACTIVE",
         "subnets":[
            "08eae331-0402-425a-923c-34f7cfe39c1b"
         ],
         "name":"private",
         "provider:physical_network":null,
         "admin_state_up":true,
         "tenant_id":"26a7980765d0414dbc1fc1f88cdb7e6e",
         "provider:network_type":"local",
         "router:external":true,
         "shared":true,
         "id":"db193ab3-96e3-4cb3-8fc5-05f4296d0324",
         "provider:segmentation_id":null
      }
   ]
}
```

Example 3.2. List networks: XML response

```
<?xml version='1.0' encoding='UTF-8'?>
<networks xmlns="http://openstack.org/quantum/api/v2.0"
    xmlns:provider="http://docs.openstack.org/ext/provider/api/v1.0"
    xmlns:quantum="http://openstack.org/quantum/api/v2.0"
    xmlns:router="http://docs.openstack.org/ext/neutron/router/api/v1.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <network>
        <status>ACTIVE</status>
        <subnets>
            <subnet>54d6f61d-db07-451c-9ab3-b9609b6b6f0b</subnet>
        </subnets>
        <name>private-network</name>
        <provider:physical_network xsi:nil="true"/>
        <admin_state_up quantum:type="bool">True</admin_state_up>
        <tenant_id>4fd44f30292945e481c7b8a0c8908869</tenant_id>
        <provider:network_type>local</provider:network_type>
        <router:external quantum:type="bool">True</router:external>
        <shared quantum:type="bool">True</shared>
        <id>d32019d3-bc6e-4319-9c1d-6722fc136a22</id>
        <provider:segmentation_id xsi:nil="true"/>
    </network>
    <network>
        <status>ACTIVE</status>
        <subnets>
            <subnet>08eae331-0402-425a-923c-34f7cfe39c1b</subnet>
        </subnets>
        <name>private</name>
        <provider:physical_network xsi:nil="true"/>
        <admin_state_up quantum:type="bool">True</admin_state_up>
        <tenant_id>26a7980765d0414dbc1fc1f88cdb7e6e</tenant_id>
        <provider:network_type>local</provider:network_type>
        <router:external quantum:type="bool">True</router:external>
        <shared quantum:type="bool">True</shared>
        <id>db193ab3-96e3-4cb3-8fc5-05f4296d0324</id>
        <provider:segmentation_id xsi:nil="true"/>
    </network>
</networks>
```

Create network

Method	URI	Description
POST	`/v2.0/networks`	Creates a network.

This operation does not require a request body. The tenant ID that you specify in the URI is the tenant that creates the network. An admin user can specify another tenant ID in the optional request body, which is the tenant who owns the network.

Normal response codes: 201

Error response codes: badRequest (400), unauthorized (401)

Request

Example 3.3. Create network: JSON request

```
{
   "network":{
      "name":"sample_network",
      "admin_state_up":true
   }
}
```

Example 3.4. Create network: XML request

```
<?xml version="1.0" encoding="UTF-8"?>
<network>
    <name>sample_network2</name>
</network>
```

Response

Example 3.5. Create network: JSON response

```
{
   "network":{
      "status":"ACTIVE",
      "subnets":[

      ],
      "name":"sample_network",
      "provider:physical_network":null,
      "admin_state_up":true,
      "tenant_id":"4fd44f30292945e481c7b8a0c8908869",
      "provider:network_type":"local",
      "shared":false,
      "id":"baed79dd-9136-4260-b9a9-d9dfa2bf6547",
      "provider:segmentation_id":null
   }
}
```

Example 3.6. Create network: XML response

```
<?xml version='1.0' encoding='UTF-8'?>
```

```
<network xmlns="http://openstack.org/quantum/api/v2.0"
    xmlns:provider="http://docs.openstack.org/ext/provider/api/v1.0"
    xmlns:quantum="http://openstack.org/quantum/api/v2.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <status>ACTIVE</status>
    <subnets quantum:type="list"/>
    <name>sample_network2</name>
    <provider:physical_network xsi:nil="true"/>
    <admin_state_up quantum:type="bool">True</admin_state_up>
    <tenant_id>4fd44f30292945e481c7b8a0c8908869</tenant_id>
    <provider:network_type>local</provider:network_type>
    <shared quantum:type="bool">False</shared>
    <id>c220b026-ece1-4ead-873f-83537f4c9f92</id>
    <provider:segmentation_id xsi:nil="true"/>
</network>
```

Bulk create networks

Method	URI	Description
POST	/v2.0/networks	Creates multiple networks in a single request.

In the request body, specify a list of networks.

The bulk create operation is always atomic. Either all or no networks in the request body are created.

Normal response codes: 201

Error response codes: badRequest (400), unauthorized (401)

Request

Example 3.7. Bulk create networks: JSON request

```
{
   "networks":[
      {
         "name":"sample_network3",
         "admin_state_up":true
      },
      {
         "name":"sample_network4",
         "admin_state_up":true
      }
   ]
}
```

Example 3.8. Bulk create networks: XML request

```
<?xml version="1.0" encoding="UTF-8"?>
<networks>
    <network>
        <name>sample_network_5</name>
    </network>
    <network>
        <name>sample_network_6</name>
    </network>
</networks>
```

Response

Example 3.9. Bulk create networks: JSON response

```
{
   "networks":[
      {
         "status":"ACTIVE",
         "subnets":[

         ],
         "name":"sample_network3",
```

```
         "provider:physical_network":null,
         "admin_state_up":true,
         "tenant_id":"4fd44f30292945e481c7b8a0c8908869",
         "provider:network_type":"local",
         "shared":false,
         "id":"bc1a76cb-8767-4c3a-bb95-018b822f2130",
         "provider:segmentation_id":null
      },
      {
         "status":"ACTIVE",
         "subnets":[

         ],
         "name":"sample_network4",
         "provider:physical_network":null,
         "admin_state_up":true,
         "tenant_id":"4fd44f30292945e481c7b8a0c8908869",
         "provider:network_type":"local",
         "shared":false,
         "id":"af374017-c9ae-4a1d-b799-ab73111476e2",
         "provider:segmentation_id":null
      }
   ]
}
```

Example 3.10. Bulk create networks: XML response

```
<?xml version='1.0' encoding='UTF-8'?>
<networks xmlns="http://openstack.org/quantum/api/v2.0"
    xmlns:provider="http://docs.openstack.org/ext/provider/api/v1.0"
    xmlns:quantum="http://openstack.org/quantum/api/v2.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <network>
        <status>ACTIVE</status>
        <subnets quantum:type="list"/>
        <name>sample_network_5</name>
        <provider:physical_network xsi:nil="true"/>
        <admin_state_up quantum:type="bool">True</admin_state_up>
        <tenant_id>4fd44f30292945e481c7b8a0c8908869</tenant_id>
        <provider:network_type>local</provider:network_type>
        <shared quantum:type="bool">False</shared>
        <id>1f370095-98f6-4079-be64-6d3d4a6adcc6</id>
        <provider:segmentation_id xsi:nil="true"/>
    </network>
    <network>
        <status>ACTIVE</status>
        <subnets quantum:type="list"/>
        <name>sample_network_6</name>
        <provider:physical_network xsi:nil="true"/>
        <admin_state_up quantum:type="bool">True</admin_state_up>
        <tenant_id>4fd44f30292945e481c7b8a0c8908869</tenant_id>
        <provider:network_type>local</provider:network_type>
        <shared quantum:type="bool">False</shared>
        <id>ee2d3158-3e80-4fb3-ba87-c99f515d85e7</id>
        <provider:segmentation_id xsi:nil="true"/>
    </network>
</networks>
```

Show network

Method	URI	Description
GET	/v2.0/networks/{network_id}	Shows information for a specified network.

You can control which attributes are returned by using the fields query parameter. For information, see Filtering and Column Selection in the *OpenStack Networking API v2.0 Reference*.

Normal response codes: 200

Error response codes: unauthorized (401), itemNotFound (404)

Request

This table shows the URI parameters for the show network request:

Name	Type	Description
{network_id}	UUID	The UUID for the network of interest to you.

This operation does not require a request body.

Response

Example 3.11. Show network: JSON response

```
{
   "network":{
      "status":"ACTIVE",
      "subnets":[
         "54d6f61d-db07-451c-9ab3-b9609b6b6f0b"
      ],
      "name":"private-network",
      "provider:physical_network":null,
      "admin_state_up":true,
      "tenant_id":"4fd44f30292945e481c7b8a0c8908869",
      "provider:network_type":"local",
      "router:external":true,
      "shared":true,
      "id":"d32019d3-bc6e-4319-9c1d-6722fc136a22",
      "provider:segmentation_id":null
   }
}
```

Example 3.12. Show network: XML response

```
<?xml version='1.0' encoding='UTF-8'?>
<network xmlns="http://openstack.org/quantum/api/v2.0"
    xmlns:provider="http://docs.openstack.org/ext/provider/api/v1.0"
    xmlns:quantum="http://openstack.org/quantum/api/v2.0"
    xmlns:router="http://docs.openstack.org/ext/neutron/router/api/v1.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <status>ACTIVE</status>
    <subnets>
```

```
        <subnet>54d6f61d-db07-451c-9ab3-b9609b6b6f0b</subnet>
    </subnets>
    <name>private-network</name>
    <provider:physical_network xsi:nil="true"/>
    <admin_state_up quantum:type="bool">True</admin_state_up>
    <tenant_id>4fd44f30292945e481c7b8a0c8908869</tenant_id>
    <provider:network_type>local</provider:network_type>
    <router:external quantum:type="bool">True</router:external>
    <shared quantum:type="bool">True</shared>
    <id>d32019d3-bc6e-4319-9c1d-6722fc136a22</id>
    <provider:segmentation_id xsi:nil="true"/>
</network>
```

Update network

Method	URI	Description
PUT	/v2.0/networks/{network_id}	Updates a specified network.

Normal response codes: 200

Error response codes: badRequest (400), unauthorized (401), forbidden (403), itemNotFound (404)

Request

This table shows the URI parameters for the update network request:

Name	Type	Description
{network_id}	UUID	The UUID for the network of interest to you.

Example 3.13. Update network: JSON request

```
{
 "network":
  {
    "name": "sample_network_5_updated"
  }
}
```

Example 3.14. Update network: XML request

```
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns="http://openstack.org/quantum/api/v2.0"
    xmlns:provider="http://docs.openstack.org/ext/provider/api/v1.0"
    xmlns:quantum="http://openstack.org/quantum/api/v2.0"
    xmlns:router="http://docs.openstack.org/ext/quantum/router/api/v1.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <name>sample-network-4-updated</name>
</network>
```

Response

Example 3.15. Update network: JSON response

```
{
   "network":{
      "status":"ACTIVE",
      "subnets":[

      ],
      "name":"sample_network_5_updated",
      "provider:physical_network":null,
      "admin_state_up":true,
      "tenant_id":"4fd44f30292945e481c7b8a0c8908869",
      "provider:network_type":"local",
      "router:external":false,
      "shared":false,
```

```
      "id":"1f370095-98f6-4079-be64-6d3d4a6adcc6",
      "provider:segmentation_id":null
   }
}
```

Example 3.16. Update network: XML response

```
<?xml version='1.0' encoding='UTF-8'?>
<network xmlns="http://openstack.org/quantum/api/v2.0"
    xmlns:provider="http://docs.openstack.org/ext/provider/api/v1.0"
    xmlns:quantum="http://openstack.org/quantum/api/v2.0"
    xmlns:router="http://docs.openstack.org/ext/neutron/router/api/v1.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <status>ACTIVE</status>
    <subnets quantum:type="list"/>
    <name>sample-network-4-updated</name>
    <provider:physical_network xsi:nil="true"/>
    <admin_state_up quantum:type="bool">True</admin_state_up>
    <tenant_id>4fd44f30292945e481c7b8a0c8908869</tenant_id>
    <provider:network_type>local</provider:network_type>
    <router:external quantum:type="bool">False</router:external>
    <shared quantum:type="bool">False</shared>
    <id>af374017-c9ae-4a1d-b799-ab73111476e2</id>
    <provider:segmentation_id xsi:nil="true"/>
</network>
```

Delete network

Method	URI	Description
DELETE	`/v2.0/networks/{network_id}`	Deletes a specified network and its associated resources.

Normal response codes: 204

Error response codes: unauthorized (401), itemNotFound (404), conflict (409)

Request

This table shows the URI parameters for the delete network request:

Name	Type	Description
`{network_id}`	UUID	The UUID for the network of interest to you.

This operation does not require a request body.

Subnets

List, show information for, create, update, and delete subnet resources.

Method	URI	Description
GET	`/v2.0/subnets`	Lists subnets to which the specified tenant has access.
POST	`/v2.0/subnets`	Creates a subnet on a specified network.
POST	`/v2.0/subnets`	Creates multiple subnets in a single request. Specify a list of subnets in the request body.
GET	`/v2.0/subnets/{subnet_id}`	Shows information for a specified subnet.
PUT	`/v2.0/subnets/{subnet_id}`	Updates a specified subnet.
DELETE	`/v2.0/subnets/{subnet_id}`	Deletes a specified subnet.

List subnets

Method	URI	Description
GET	/v2.0/subnets	Lists subnets to which the specified tenant has access.

Default policy settings returns exclusively subnets owned by the tenant submitting the request, unless the request is submitted by an user with administrative rights. You can control which attributes are returned by using the fields query parameter. You can filter results by using query string parameters. For information, see Filtering and Column Selection in the *OpenStack Networking API v2.0 Reference*.

Normal response codes: 200

Error response codes: unauthorized (401)

Request

This operation does not require a request body.

Response

Example 3.17. List subnets: JSON response

```
{
   "subnets":[
      {
         "name":"private-subnet",
         "enable_dhcp":true,
         "network_id":"db193ab3-96e3-4cb3-8fc5-05f4296d0324",
         "tenant_id":"26a7980765d0414dbc1fc1f88cdb7e6e",
         "dns_nameservers":[

         ],
         "allocation_pools":[
            {
               "start":"10.0.0.2",
               "end":"10.0.0.254"
            }
         ],
         "host_routes":[

         ],
         "ip_version":4,
         "gateway_ip":"10.0.0.1",
         "cidr":"10.0.0.0/24",
         "id":"08eae331-0402-425a-923c-34f7cfe39c1b"
      },
      {
         "name":"my_subnet",
         "enable_dhcp":true,
         "network_id":"d32019d3-bc6e-4319-9c1d-6722fc136a22",
         "tenant_id":"4fd44f30292945e481c7b8a0c8908869",
         "dns_nameservers":[

         ],
```

```
        "allocation_pools":[
           {
              "start":"192.0.0.2",
              "end":"192.255.255.254"
           }
        ],
        "host_routes":[

        ],
        "ip_version":4,
        "gateway_ip":"192.0.0.1",
        "cidr":"192.0.0.0/8",
        "id":"54d6f61d-db07-451c-9ab3-b9609b6b6f0b"
     }
  ]
}
```

Example 3.18. List subnets: XML response

```
<?xml version='1.0' encoding='UTF-8'?>
<subnets xmlns="http://openstack.org/quantum/api/v2.0"
    xmlns:quantum="http://openstack.org/quantum/api/v2.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <subnet>
        <name>private-subnet</name>
        <enable_dhcp quantum:type="bool">True</enable_dhcp>
        <network_id>db193ab3-96e3-4cb3-8fc5-05f4296d0324</network_id>
        <tenant_id>26a7980765d0414dbc1fc1f88cdb7e6e</tenant_id>
        <dns_nameservers quantum:type="list"/>
        <allocation_pools>
            <allocation_pool>
                <start>10.0.0.2</start>
                <end>10.0.0.254</end>
            </allocation_pool>
        </allocation_pools>
        <host_routes quantum:type="list"/>
        <ip_version quantum:type="long">4</ip_version>
        <gateway_ip>10.0.0.1</gateway_ip>
        <cidr>10.0.0.0/24</cidr>
        <id>08eae331-0402-425a-923c-34f7cfe39c1b</id>
    </subnet>
    <subnet>
        <name>my_subnet</name>
        <enable_dhcp quantum:type="bool">True</enable_dhcp>
        <network_id>d32019d3-bc6e-4319-9c1d-6722fc136a22</network_id>
        <tenant_id>4fd44f30292945e481c7b8a0c8908869</tenant_id>
        <dns_nameservers quantum:type="list"/>
        <allocation_pools>
            <allocation_pool>
                <start>192.0.0.2</start>
                <end>192.255.255.254</end>
            </allocation_pool>
        </allocation_pools>
        <host_routes quantum:type="list"/>
        <ip_version quantum:type="long">4</ip_version>
        <gateway_ip>192.0.0.1</gateway_ip>
        <cidr>192.0.0.0/8</cidr>
        <id>54d6f61d-db07-451c-9ab3-b9609b6b6f0b</id>
    </subnet>
</subnets>
```

This operation does not return a response body.

Create subnet

Method	URI	Description
POST	/v2.0/subnets	Creates a subnet on a specified network.

By default, OpenStack Networking creates IP v4 subnets. To create an IP v6 subnet, you must specify the value 6 for the `ip_version` attribute in the request body. OpenStack Networking does not try to derive the correct IP version from the provided CIDR. If the parameter for the gateway address, `gateway_ip`, is not specified, OpenStack Networking allocates an address from the cidr for the gateway for the subnet.

To specify a subnet without a gateway, specify the value null for the `gateway_ip` attribute in the request body. If allocation pools attribute, `allocation_pools`, is not specified, OpenStack Networking automatically allocates pools for covering all IP addresses in the CIDR, excluding the address reserved for the subnet gateway. Otherwise, you can explicitly specify allocation pools as shown in the following example.

When `allocation_pools` and `gateway_ip` are both specified, it is up to the user to ensure that the gateway IP does not overlap with the specified allocation pools; otherwise a 409 Conflict error occurs.

Normal response codes: 201

Error response codes: badRequest (400), unauthorized (401), forbidden (403), itemNotFound (404), conflict (409)

Request

Example 3.19. Create subnet: JSON request

```
{
   "subnet":{
      "subnet":{
         "network_id":"d32019d3-bc6e-4319-9c1d-6722fc136a22",
         "ip_version":4,
         "cidr":"192.168.199.0/24"
      }
   }
}
```

Example 3.20. Create subnet: XML request

```
<?xml version="1.0" encoding="UTF-8"?>
<subnet>
    <name>test_subnet_1</name>
    <network_id>d32019d3-bc6e-4319-9c1d-6722fc136a22</network_id>
    <cidr>192.0.0.0/8</cidr>
    <ip_version>4</ip_version>
</subnet>
```

This operation does not require a request body.

Response

Example 3.21. Create subnet: JSON response

```
{
   "subnet":{
      "name":"",
      "enable_dhcp":true,
      "network_id":"d32019d3-bc6e-4319-9c1d-6722fc136a22",
      "tenant_id":"4fd44f30292945e481c7b8a0c8908869",
      "dns_nameservers":[

      ],
      "allocation_pools":[
         {
            "start":"192.168.199.2",
            "end":"192.168.199.254"
         }
      ],
      "host_routes":[

      ],
      "ip_version":4,
      "gateway_ip":"192.168.199.1",
      "cidr":"192.168.199.0/24",
      "id":"3b80198d-4f7b-4f77-9ef5-774d54e17126"
   }
}
```

Example 3.22. Create subnet: XML response

```
<?xml version='1.0' encoding='UTF-8'?>
<subnet xmlns="http://openstack.org/quantum/api/v2.0"
    xmlns:quantum="http://openstack.org/quantum/api/v2.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <name>test_subnet_1</name>
    <enable_dhcp quantum:type="bool">True</enable_dhcp>
    <network_id>d32019d3-bc6e-4319-9c1d-6722fc136a22</network_id>
    <tenant_id>4fd44f30292945e481c7b8a0c8908869</tenant_id>
    <dns_nameservers quantum:type="list"/>
    <allocation_pools>
        <allocation_pool>
            <start>192.0.0.2</start>
            <end>192.255.255.254</end>
        </allocation_pool>
    </allocation_pools>
    <host_routes quantum:type="list"/>
    <ip_version quantum:type="int">4</ip_version>
    <gateway_ip>192.0.0.1</gateway_ip>
    <cidr>192.0.0.0/8</cidr>
    <id>54d6f61d-db07-451c-9ab3-b9609b6b6f0b</id>
</subnet>
```

This operation does not return a response body.

Bulk create subnet

Method	URI	Description
POST	`/v2.0/subnets`	Creates multiple subnets in a single request. Specify a list of subnets in the request body.

The bulk create operation is always atomic. Either all or no subnets in the request body are created.

Normal response codes: 201

Error response codes: badRequest (400), unauthorized (401), forbidden (403), itemNotFound (404), conflict (409)

Request

Example 3.23. Bulk create subnet: JSON request

```
{
   "subnets":[
      {
         "cidr":"192.168.199.0/24",
         "ip_version":4,
         "network_id":"e6031bc2-901a-4c66-82da-f4c32ed89406"
      },
      {
         "cidr":"10.56.4.0/22",
         "ip_version":4,
         "network_id":"64239a54-dcc4-4b39-920b-b37c2144effa"
      }
   ]
}
```

Example 3.24. Bulk create subnet: XML request

```
<?xml version="1.0" encoding="UTF-8"?>
<subnets>
    <subnet>
        <name>test_subnet_1</name>
        <network_id>a3775a7d-9f8b-4148-be81-c84bbd0837ce</network_id>
        <cidr>10.0.0.0/8</cidr>
        <ip_version>4</ip_version>
    </subnet>
    <subnet>
        <name>test_subnet_2</name>
        <network_id>a3775a7d-9f8b-4148-be81-c84bbd0837ce</network_id>
        <cidr>192.168.0.0/16</cidr>
        <ip_version>4</ip_version>
    </subnet>
</subnets>
```

This operation does not require a request body.

Response

Example 3.25. Bulk create subnet: JSON response


```
{
   "subnets":[
      {
         "allocation_pools":[
            {
               "end":"192.168.199.254",
               "start":"192.168.199.2"
            }
         ],
         "cidr":"192.168.199.0/24",
         "dns_nameservers":[

         ],
         "enable_dhcp":true,
         "gateway_ip":"192.168.199.1",
         "host_routes":[

         ],
         "id":"0468a7a7-290d-4127-aedd-6c9449775a24",
         "ip_version":4,
         "name":"",
         "network_id":"e6031bc2-901a-4c66-82da-f4c32ed89406",
         "tenant_id":"d19231fc08ec4bc4829b668040d34512"
      },
      {
         "allocation_pools":[
            {
               "end":"10.56.7.254",
               "start":"10.56.4.2"
            }
         ],
         "cidr":"10.56.4.0/22",
         "dns_nameservers":[

         ],
         "enable_dhcp":true,
         "gateway_ip":"10.56.4.1",
         "host_routes":[

         ],
         "id":"b0e7435c-1512-45fb-aa9e-9a7c5932fb30",
         "ip_version":4,
         "name":"",
         "network_id":"64239a54-dcc4-4b39-920b-b37c2144effa",
         "tenant_id":"d19231fc08ec4bc4829b668040d34512"
      }
   ]
}
```


Example 3.26. Bulk create subnet: XML response


```
<?xml version='1.0' encoding='UTF-8'?>
<subnets xmlns="http://openstack.org/quantum/api/v2.0"
    xmlns:quantum="http://openstack.org/quantum/api/v2.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
```

```
    <subnet>
        <name>test_subnet_1</name>
        <enable_dhcp quantum:type="bool">True</enable_dhcp>
        <network_id>a3775a7d-9f8b-4148-be81-c84bbd0837ce</network_id>
        <tenant_id>60cd4f6dbc2f491982a284e7b83b5be3</tenant_id>
        <dns_nameservers quantum:type="list"/>
        <allocation_pools>
            <allocation_pool>
                <start>10.0.0.2</start>
                <end>10.255.255.254</end>
            </allocation_pool>
        </allocation_pools>
        <host_routes quantum:type="list"/>
        <ip_version quantum:type="int">4</ip_version>
        <gateway_ip>10.0.0.1</gateway_ip>
        <cidr>10.0.0.0/8</cidr>
        <id>bd3fd365-fe19-431a-be63-07017a09316c</id>
    </subnet>
    <subnet>
        <name>test_subnet_2</name>
        <enable_dhcp quantum:type="bool">True</enable_dhcp>
        <network_id>a3775a7d-9f8b-4148-be81-c84bbd0837ce</network_id>
        <tenant_id>60cd4f6dbc2f491982a284e7b83b5be3</tenant_id>
        <dns_nameservers quantum:type="list"/>
        <allocation_pools>
            <allocation_pool>
                <start>192.168.0.2</start>
                <end>192.168.255.254</end>
            </allocation_pool>
        </allocation_pools>
        <host_routes quantum:type="list"/>
        <ip_version quantum:type="int">4</ip_version>
        <gateway_ip>192.168.0.1</gateway_ip>
        <cidr>192.168.0.0/16</cidr>
        <id>86e7c838-fb75-402b-9dbf-d68166e3f5fe</id>
    </subnet>
</subnets>
```

This operation does not return a response body.

Show subnet

Method	URI	Description
GET	/v2.0/subnets/{subnet_id}	Shows information for a specified subnet.

You can control which attributes are returned by using the fields query parameter. For information, see Filtering and Column Selection in the *OpenStack Networking API v2.0 Reference*.

Normal response codes: 201

Error response codes: unauthorized (401), itemNotFound (404)

Request

This table shows the URI parameters for the show subnet request:

Name	Type	Description
{subnet_id}	UUID	The UUID for the subnet of interest to you.

This operation does not require a request body.

Response

Example 3.27. Show subnet: JSON response

```
{
   "subnet":{
      "name":"my_subnet",
      "enable_dhcp":true,
      "network_id":"d32019d3-bc6e-4319-9c1d-6722fc136a22",
      "tenant_id":"4fd44f30292945e481c7b8a0c8908869",
      "dns_nameservers":[

      ],
      "allocation_pools":[
         {
            "start":"192.0.0.2",
            "end":"192.255.255.254"
         }
      ],
      "host_routes":[

      ],
      "ip_version":4,
      "gateway_ip":"192.0.0.1",
      "cidr":"192.0.0.0/8",
      "id":"54d6f61d-db07-451c-9ab3-b9609b6b6f0b"
   }
}
```

Example 3.28. Show subnet: XML response

```
<?xml version='1.0' encoding='UTF-8'?>
```

```
<subnet xmlns="http://openstack.org/quantum/api/v2.0"
    xmlns:quantum="http://openstack.org/quantum/api/v2.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <name>test_subnet_1</name>
    <enable_dhcp quantum:type="bool">True</enable_dhcp>
    <network_id>d32019d3-bc6e-4319-9c1d-6722fc136a22</network_id>
    <tenant_id>4fd44f30292945e481c7b8a0c8908869</tenant_id>
    <dns_nameservers quantum:type="list"/>
    <allocation_pools>
        <allocation_pool>
            <start>192.0.0.2</start>
            <end>192.255.255.254</end>
        </allocation_pool>
    </allocation_pools>
    <host_routes quantum:type="list"/>
    <ip_version quantum:type="long">4</ip_version>
    <gateway_ip>192.0.0.1</gateway_ip>
    <cidr>192.0.0.0/8</cidr>
    <id>54d6f61d-db07-451c-9ab3-b9609b6b6f0b</id>
</subnet>
```

This operation does not return a response body.

Update subnet

Method	URI	Description
PUT	/v2.0/subnets/{subnet_id}	Updates a specified subnet.

Some attributes, such as IP version (ip_version), CIDR (cidr), and IP allocation pools (allocation_pools) cannot be updated. Attempting to update these attributes results in a `400 Bad Request` error.

Normal response codes: 201

Error response codes: badRequest (400), unauthorized (401), forbidden (403), itemNotFound (404)

Request

This table shows the URI parameters for the update subnet request:

Name	Type	Description
{subnet_id}	UUID	The UUID for the subnet of interest to you.

Example 3.29. Update subnet: JSON request

```
{
   "subnet":{
      "subnet":{
         "name":"my_subnet"
      }
   }
}
```

Example 3.30. Update subnet: XML request

```
<?xml version="1.0" encoding="UTF-8"?>
<subnet>
    <name>my_subnet</name>
</subnet>
```

This operation does not require a request body.

Response

Example 3.31. Update subnet: JSON response

```
{
   "subnet":{
      "name":"private-subnet",
      "enable_dhcp":true,
      "network_id":"db193ab3-96e3-4cb3-8fc5-05f4296d0324",
      "tenant_id":"26a7980765d0414dbc1fc1f88cdb7e6e",
      "dns_nameservers":[

      ],
      "allocation_pools":[
```

```
         {
            "start":"10.0.0.2",
            "end":"10.0.0.254"
         }
      ],
      "host_routes":[

      ],
      "ip_version":4,
      "gateway_ip":"10.0.0.1",
      "cidr":"10.0.0.0/24",
      "id":"08eae331-0402-425a-923c-34f7cfe39c1b"
   }
}
```

Example 3.32. Update subnet: XML response

```
<?xml version='1.0' encoding='UTF-8'?>
<subnet xmlns="http://openstack.org/quantum/api/v2.0"
    xmlns:quantum="http://openstack.org/quantum/api/v2.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <name>my_subnet</name>
    <enable_dhcp quantum:type="bool">True</enable_dhcp>
    <network_id>d32019d3-bc6e-4319-9c1d-6722fc136a22</network_id>
    <tenant_id>4fd44f30292945e481c7b8a0c8908869</tenant_id>
    <dns_nameservers quantum:type="list"/>
    <allocation_pools>
        <allocation_pool>
            <start>192.0.0.2</start>
            <end>192.255.255.254</end>
        </allocation_pool>
    </allocation_pools>
    <host_routes quantum:type="list"/>
    <ip_version quantum:type="long">4</ip_version>
    <gateway_ip>192.0.0.1</gateway_ip>
    <cidr>192.0.0.0/8</cidr>
    <id>54d6f61d-db07-451c-9ab3-b9609b6b6f0b</id>
</subnet>
```

This operation does not return a response body.

Delete subnet

Method	URI	Description
DELETE	`/v2.0/subnets/{subnet_id}`	Deletes a specified subnet.

The operation fails if subnet IP addresses are still allocated.

Normal response codes: 204

Error response codes: unauthorized (401), itemNotFound (404), conflict (409)

Request

This table shows the URI parameters for the delete subnet request:

Name	Type	Description
`{subnet_id}`	UUID	The UUID for the subnet of interest to you.

This operation does not require a request body.

Ports

List, show information for, create, update, and delete ports.

Method	URI	Description
GET	`/v2.0/ports`	Lists ports to which the tenant has access.
POST	`/v2.0/ports`	Creates a port on a specified network.
POST	`/v2.0/ports`	Creates multiple ports in a single request. Specify a list of ports in the request body.
GET	`/v2.0/ports/{port_id}`	Shows information for a specified port.
PUT	`/v2.0/ports/{port_id}`	Updates a specified port.
DELETE	`/v2.0/ports/{port_id}`	Deletes a specified port.

List ports

Method	URI	Description
GET	/v2.0/ports	Lists ports to which the tenant has access.

Default policy settings return only those subnets that are owned by the tenant who submits the request, unless the request is submitted by an user with administrative rights. Users can control which attributes are returned by using the fields query parameter. Additionally, you can filter results by using query string parameters. For information, see Filtering and Column Selection in the *OpenStack Networking API v2.0 Reference*.

Normal response codes: 200

Error response codes: unauthorized (401)

Request

This operation does not require a request body.

Response

Example 3.33. List ports: JSON response

```
{
   "ports":[
      {
         "status":"ACTIVE",
         "binding:host_id":"devstack-havana",
         "name":"vip-a54bc6e7-2e28-4c55-a676-6146a4c0f8b9",
         "allowed_address_pairs":[

         ],
         "admin_state_up":true,
         "network_id":"d32019d3-bc6e-4319-9c1d-6722fc136a22",
         "tenant_id":"4fd44f30292945e481c7b8a0c8908869",
         "extra_dhcp_opts":[

         ],
         "binding:vif_type":"ovs",
         "device_owner":"neutron:LOADBALANCER",
         "binding:capabilities":{
            "port_filter":true
         },
         "mac_address":"fa:16:3e:47:57:a0",
         "fixed_ips":[
            {
               "subnet_id":"54d6f61d-db07-451c-9ab3-b9609b6b6f0b",
               "ip_address":"192.0.0.3"
            }
         ],
         "id":"36242e87-0bae-49d7-bc59-a0867476c69a",
         "security_groups":[
            "d30c3c54-5dba-49cf-a323-48a86f078d2d"
         ],
         "device_id":"56016959-08ad-566c-8533-6240aff17dd5"
      },
      {
```

```
        "status":"DOWN",
        "binding:host_id":"",
        "name":"my_port",
        "allowed_address_pairs":[

        ],
        "admin_state_up":true,
        "network_id":"d32019d3-bc6e-4319-9c1d-6722fc136a22",
        "tenant_id":"4fd44f30292945e481c7b8a0c8908869",
        "extra_dhcp_opts":[

        ],
        "binding:vif_type":"unbound",
        "device_owner":"",
        "binding:capabilities":{
           "port_filter":false
        },
        "mac_address":"fa:16:3e:6c:e8:35",
        "fixed_ips":[

        ],
        "id":"41064069-24d6-46e8-9b5a-6da327e357b3",
        "security_groups":[
           "d30c3c54-5dba-49cf-a323-48a86f078d2d"
        ],
        "device_id":""
     },
     {
        "status":"DOWN",
        "binding:host_id":"",
        "name":"",
        "allowed_address_pairs":[

        ],
        "admin_state_up":true,
        "network_id":"ee2d3158-3e80-4fb3-ba87-c99f515d85e7",
        "tenant_id":"4fd44f30292945e481c7b8a0c8908869",
        "extra_dhcp_opts":[

        ],
        "binding:vif_type":"unbound",
        "device_owner":"",
        "binding:capabilities":{
           "port_filter":false
        },
        "mac_address":"fa:16:3e:80:14:5b",
        "fixed_ips":[

        ],
        "id":"6f9f6319-ce4b-4267-a5f8-558d6795632d",
        "security_groups":[
           "d30c3c54-5dba-49cf-a323-48a86f078d2d"
        ],
        "device_id":""
     },
     {
        "status":"ACTIVE",
        "binding:host_id":"devstack-havana",
        "name":"",
        "allowed_address_pairs":[
```

```
        ],
        "admin_state_up":true,
        "network_id":"d32019d3-bc6e-4319-9c1d-6722fc136a22",
        "tenant_id":"4fd44f30292945e481c7b8a0c8908869",
        "extra_dhcp_opts":[

        ],
        "binding:vif_type":"ovs",
        "device_owner":"network:dhcp",
        "binding:capabilities":{
           "port_filter":true
        },
        "mac_address":"fa:16:3e:3b:63:e8",
        "fixed_ips":[
           {
              "subnet_id":"54d6f61d-db07-451c-9ab3-b9609b6b6f0b",
              "ip_address":"192.0.0.2"
           }
        ],
        "id":"9842e9ab-7849-4bb5-8441-9fa223bfce45",
        "security_groups":[

        ],
        "device_id":"dhcp56016959-08ad-566c-8533-6240aff17dd5-d32019d3-
bc6e-4319-9c1d-6722fc136a22"
     },
     {
        "status":"ACTIVE",
        "binding:host_id":"devstack-havana",
        "name":"",
        "allowed_address_pairs":[

        ],
        "admin_state_up":true,
        "network_id":"db193ab3-96e3-4cb3-8fc5-05f4296d0324",
        "tenant_id":"26a7980765d0414dbc1fc1f88cdb7e6e",
        "extra_dhcp_opts":[

        ],
        "binding:vif_type":"ovs",
        "device_owner":"network:dhcp",
        "binding:capabilities":{
           "port_filter":true
        },
        "mac_address":"fa:16:3e:f7:80:62",
        "fixed_ips":[
           {
              "subnet_id":"08eae331-0402-425a-923c-34f7cfe39c1b",
              "ip_address":"10.0.0.3"
           }
        ],
        "id":"d2159251-552f-47ae-9960-f80d2aa6864f",
        "security_groups":[

        ],
        "device_id":"dhcp56016959-08ad-566c-8533-6240aff17dd5-
db193ab3-96e3-4cb3-8fc5-05f4296d0324"
     }
  ]
```

```
}
```

Example 3.34. List ports: XML response

```
<?xml version='1.0' encoding='UTF-8'?>
<ports xmlns="http://openstack.org/quantum/api/v2.0" xmlns:binding="http://
docs.openstack.org/ext/binding/api/v1.0" xmlns:quantum="http://openstack.org/
quantum/api/v2.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <port>
        <status>ACTIVE</status>
        <name />
        <admin_state_up quantum:type="bool">True</admin_state_up>
        <network_id>ebda9658-093b-41ba-80ce-0cf8cb8365d4</network_id>
        <tenant_id>63878e4c5dd649d2a980e37aefddfa87</tenant_id>
        <binding:vif_type>ovs</binding:vif_type>
        <device_owner>compute:None</device_owner>
        <binding:capabilities>
            <port_filter quantum:type="bool">False</port_filter>
        </binding:capabilities>
        <mac_address>fa:16:3e:b9:ef:05</mac_address>
        <fixed_ips>
            <fixed_ip>
                <subnet_id>aca4d43c-c48c-4a2c-9bb6-ba374ef7e135</subnet_id>
                <ip_address>172.24.4.227</ip_address>
            </fixed_ip>
        </fixed_ips>
        <id>664ebd1a-facd-4c20-948c-07a784475ab0</id>
        <device_id>f288bb5f-920d-4276-8345-2c0319c16f58</device_id>
    </port>
    <port>
        <status>DOWN</status>
        <name />
        <admin_state_up quantum:type="bool">True</admin_state_up>
        <network_id>ebda9658-093b-41ba-80ce-0cf8cb8365d4</network_id>
        <tenant_id />
        <binding:vif_type>ovs</binding:vif_type>
        <device_owner>network:router_gateway</device_owner>
        <binding:capabilities>
            <port_filter quantum:type="bool">False</port_filter>
        </binding:capabilities>
        <mac_address>fa:16:3e:4a:3a:a2</mac_address>
        <fixed_ips>
            <fixed_ip>
                <subnet_id>aca4d43c-c48c-4a2c-9bb6-ba374ef7e135</subnet_id>
                <ip_address>172.24.4.226</ip_address>
            </fixed_ip>
        </fixed_ips>
        <id>c5ca7017-c390-4ccc-8cd7-333747e57fef</id>
        <device_id>0dc517bf-9169-4aa6-88b7-569219962881</device_id>
    </port>
    <port>
        <status>ACTIVE</status>
        <name />
        <admin_state_up quantum:type="bool">True</admin_state_up>
        <network_id>9d83c053-b0a4-4682-ae80-c00df269ce0a</network_id>
        <tenant_id>625887121e364204873d362b553ab171</tenant_id>
        <binding:vif_type>ovs</binding:vif_type>
        <device_owner>network:router_interface</device_owner>
        <binding:capabilities>
            <port_filter quantum:type="bool">False</port_filter>
```

```
        </binding:capabilities>
        <mac_address>fa:16:3e:2d:dc:7e</mac_address>
        <fixed_ips>
            <fixed_ip>
                <subnet_id>a318fcb4-9ff0-4485-b78c-9e6738c21b26</subnet_id>
                <ip_address>10.0.0.1</ip_address>
            </fixed_ip>
        </fixed_ips>
        <id>d7815f5b-a228-47bb-a5e5-f139c4e476f6</id>
        <device_id>0dc517bf-9169-4aa6-88b7-569219962881</device_id>
    </port>
    <port>
        <status>ACTIVE</status>
        <name />
        <admin_state_up quantum:type="bool">True</admin_state_up>
        <network_id>9d83c053-b0a4-4682-ae80-c00df269ce0a</network_id>
        <tenant_id>625887121e364204873d362b553ab171</tenant_id>
        <binding:vif_type>ovs</binding:vif_type>
        <device_owner>network:dhcp</device_owner>
        <binding:capabilities>
            <port_filter quantum:type="bool">False</port_filter>
        </binding:capabilities>
        <mac_address>fa:16:3e:73:6d:1c</mac_address>
        <fixed_ips>
            <fixed_ip>
                <subnet_id>a318fcb4-9ff0-4485-b78c-9e6738c21b26</subnet_id>
                <ip_address>10.0.0.2</ip_address>
            </fixed_ip>
        </fixed_ips>
        <id>f8639521-fab2-4879-94b2-83a47bee8a26</id>
        <device_id>dhcpe1b8334f-9be9-5e49-aeee-b31e6df6c847-9d83c053-
b0a4-4682-ae80-c00df269ce0a</device_id>
    </port>
</ports>
```

This operation does not return a response body.

Create port

Method	URI	Description
POST	/v2.0/ports	Creates a port on a specified network.

You must specify the network where the port is to created in the network_id attribute in the request body.

Normal response codes: 201

Error response codes: badRequest (400), unauthorized (401), forbidden (403), itemNotFound (404), macGenerationFailure (503), serviceUnavailable (503)

Request

Example 3.35. Create port: JSON request

```
{
   "port":{
      "network_id":"ee2d3158-3e80-4fb3-ba87-c99f515d85e7",
      "admin_state_up":true
   }
}
```

Response

Example 3.36. Create port: JSON response

```
{
   "port":{
      "status":"DOWN",
      "binding:host_id":"",
      "name":"",
      "allowed_address_pairs":[

      ],
      "admin_state_up":true,
      "network_id":"ee2d3158-3e80-4fb3-ba87-c99f515d85e7",
      "tenant_id":"4fd44f30292945e481c7b8a0c8908869",
      "binding:vif_type":"unbound",
      "device_owner":"",
      "binding:capabilities":{
         "port_filter":false
      },
      "mac_address":"fa:16:3e:80:14:5b",
      "fixed_ips":[

      ],
      "id":"6f9f6319-ce4b-4267-a5f8-558d6795632d",
      "security_groups":[
         "d30c3c54-5dba-49cf-a323-48a86f078d2d"
      ],
      "device_id":""
   }
}
```

Example 3.37. Create port: XML response

```
<?xml version="1.0" encoding="UTF-8"?>
<port xmlns="http://openstack.org/quantum/api/v2.0"
    xmlns:binding="http://docs.openstack.org/ext/binding/api/v1.0"
    xmlns:quantum="http://openstack.org/quantum/api/v2.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <status>DOWN</status>
    <binding:host_id xsi:nil="true"/>
    <name>test_port_1</name>
    <admin_state_up quantum:type="bool">True</admin_state_up>
    <network_id>a3775a7d-9f8b-4148-be81-c84bbd0837ce</network_id>
    <tenant_id>60cd4f6dbc2f491982a284e7b83b5be3</tenant_id>
    <binding:vif_type>ovs</binding:vif_type>
    <device_owner/>
    <binding:capabilities>
        <port_filter quantum:type="bool">True</port_filter>
    </binding:capabilities>
    <mac_address>fa:16:3e:c9:8d:cf</mac_address>
    <fixed_ips quantum:type="list"/>
    <id>7f0aa3f1-883a-43b2-8d1b-e85fac52b417</id>
    <security_groups>
        <security_group>99f465bc-0d7c-4142-8829-7ae0179f2fa8</security_group>
    </security_groups>
    <device_id/>
</port>
```

This operation does not return a response body.

Bulk create ports

Method	URI	Description
POST	`/v2.0/ports`	Creates multiple ports in a single request. Specify a list of ports in the request body.

Guarantees the atomic completion of the bulk operation.

Normal response codes: 201

Error response codes: badRequest (400), unauthorized (401), forbidden (403), itemNotFound (404), conflict (409), macGenerationFailure (503)

Request

Example 3.38. Bulk create ports: JSON request

```
{
   "ports":[
      {
         "name":"sample_port_1",
         "admin_state_up":false,
         "network_id":"a3775a7d-9f8b-4148-be81-c84bbd0837ce"
      },
      {
         "name":"sample_port_2",
         "admin_state_up":false,
         "network_id":"a3775a7d-9f8b-4148-be81-c84bbd0837ce"
      }
   ]
}
```

Example 3.39. Bulk create ports: XML request

```
<?xml version="1.0" encoding="UTF-8"?>
<ports>
    <port>
        <name>test_port_1</name>
        <network_id>a3775a7d-9f8b-4148-be81-c84bbd0837ce</network_id>
    </port>
    <port>
        <name>test_port_2</name>
        <network_id>a3775a7d-9f8b-4148-be81-c84bbd0837ce</network_id>
    </port>
</ports>
```

This operation does not require a request body.

Response

Example 3.40. Bulk create ports: JSON response

```
{
   "ports":[
      {
```

```
         "status":"DOWN",
         "binding:host_id":null,
         "name":"sample_port_1",
         "admin_state_up":true,
         "network_id":"a3775a7d-9f8b-4148-be81-c84bbd0837ce",
         "tenant_id":"60cd4f6dbc2f491982a284e7b83b5be3",
         "binding:vif_type":"ovs",
         "device_owner":"",
         "binding:capabilities":{
            "port_filter":true
         },
         "mac_address":"fa:16:3e:2e:7c:8a",
         "fixed_ips":[

         ],
         "id":"8fb361d8-bab0-418d-b1b8-7204a230fb06",
         "security_groups":[
            "99f465bc-0d7c-4142-8829-7ae0179f2fa8"
         ],
         "device_id":""
      },
      {
         "status":"DOWN",
         "binding:host_id":null,
         "name":"sample_port_2",
         "admin_state_up":false,
         "network_id":"a3775a7d-9f8b-4148-be81-c84bbd0837ce",
         "tenant_id":"60cd4f6dbc2f491982a284e7b83b5be3",
         "binding:vif_type":"ovs",
         "device_owner":"",
         "binding:capabilities":{
            "port_filter":true
         },
         "mac_address":"fa:16:3e:0a:4e:13",
         "fixed_ips":[

         ],
         "id":"d4c93b0b-f593-424e-a199-d648478a5a3c",
         "security_groups":[
            "99f465bc-0d7c-4142-8829-7ae0179f2fa8"
         ],
         "device_id":""
      }
   ]
}
```

Example 3.41. Bulk create ports: XML response

```
<?xml version='1.0' encoding='UTF-8'?>
<ports xmlns="http://openstack.org/quantum/api/v2.0"
    xmlns:binding="http://docs.openstack.org/ext/binding/api/v1.0"
    xmlns:quantum="http://openstack.org/quantum/api/v2.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <port>
        <status>DOWN</status>
        <binding:host_id xsi:nil="true"/>
        <name>test_port_1</name>
        <admin_state_up quantum:type="bool">True</admin_state_up>
        <network_id>a3775a7d-9f8b-4148-be81-c84bbd0837ce</network_id>
        <tenant_id>60cd4f6dbc2f491982a284e7b83b5be3</tenant_id>
```

```
        <binding:vif_type>ovs</binding:vif_type>
        <device_owner/>
        <binding:capabilities>
            <port_filter quantum:type="bool">True</port_filter>
        </binding:capabilities>
        <mac_address>fa:16:3e:c9:8d:cf</mac_address>
        <fixed_ips quantum:type="list"/>
        <id>7f0aa3f1-883a-43b2-8d1b-e85fac52b417</id>
        <security_groups>
            <security_group>99f465bc-0d7c-4142-8829-7ae0179f2fa8</
security_group>
        </security_groups>
        <device_id/>
    </port>
    <port>
        <status>DOWN</status>
        <binding:host_id xsi:nil="true"/>
        <name>test_port_2</name>
        <admin_state_up quantum:type="bool">True</admin_state_up>
        <network_id>a3775a7d-9f8b-4148-be81-c84bbd0837ce</network_id>
        <tenant_id>60cd4f6dbc2f491982a284e7b83b5be3</tenant_id>
        <binding:vif_type>ovs</binding:vif_type>
        <device_owner/>
        <binding:capabilities>
            <port_filter quantum:type="bool">True</port_filter>
        </binding:capabilities>
        <mac_address>fa:16:3e:79:90:81</mac_address>
        <fixed_ips quantum:type="list"/>
        <id>a4a81484-c1c4-4b2b-95bc-f8c4484241d0</id>
        <security_groups>
            <security_group>99f465bc-0d7c-4142-8829-7ae0179f2fa8</
security_group>
        </security_groups>
        <device_id/>
    </port>
</ports>
```

This operation does not return a response body.

Show port

Method	URI	Description
GET	/v2.0/ports/{port_id}	Shows information for a specified port.

Normal response codes: 200

Error response codes: unauthorized (401), itemNotFound (404)

Request

This table shows the URI parameters for the show port request:

Name	Type	Description
{port_id}	UUID	The UUID for the port of interest to you.

This operation does not require a request body.

Response

Example 3.42. Show port: JSON response

```
{
   "ports":[
      {
         "status":"ACTIVE",
         "binding:host_id":"devstack-havana",
         "name":"vip-a54bc6e7-2e28-4c55-a676-6146a4c0f8b9",
         "allowed_address_pairs":[

         ],
         "admin_state_up":true,
         "network_id":"d32019d3-bc6e-4319-9c1d-6722fc136a22",
         "tenant_id":"4fd44f30292945e481c7b8a0c8908869",
         "extra_dhcp_opts":[

         ],
         "binding:vif_type":"ovs",
         "device_owner":"neutron:LOADBALANCER",
         "binding:capabilities":{
            "port_filter":true
         },
         "mac_address":"fa:16:3e:47:57:a0",
         "fixed_ips":[
            {
               "subnet_id":"54d6f61d-db07-451c-9ab3-b9609b6b6f0b",
               "ip_address":"192.0.0.3"
            }
         ],
         "id":"36242e87-0bae-49d7-bc59-a0867476c69a",
         "security_groups":[
            "d30c3c54-5dba-49cf-a323-48a86f078d2d"
         ],
         "device_id":"56016959-08ad-566c-8533-6240aff17dd5"
      },
      {
         "status":"DOWN",
```

```
        "binding:host_id":"",
        "name":"my_port",
        "allowed_address_pairs":[

        ],
        "admin_state_up":true,
        "network_id":"d32019d3-bc6e-4319-9c1d-6722fc136a22",
        "tenant_id":"4fd44f30292945e481c7b8a0c8908869",
        "extra_dhcp_opts":[

        ],
        "binding:vif_type":"unbound",
        "device_owner":"",
        "binding:capabilities":{
           "port_filter":false
        },
        "mac_address":"fa:16:3e:6c:e8:35",
        "fixed_ips":[

        ],
        "id":"41064069-24d6-46e8-9b5a-6da327e357b3",
        "security_groups":[
           "d30c3c54-5dba-49cf-a323-48a86f078d2d"
        ],
        "device_id":""
     },
     {
        "status":"DOWN",
        "binding:host_id":"",
        "name":"",
        "allowed_address_pairs":[

        ],
        "admin_state_up":true,
        "network_id":"ee2d3158-3e80-4fb3-ba87-c99f515d85e7",
        "tenant_id":"4fd44f30292945e481c7b8a0c8908869",
        "extra_dhcp_opts":[

        ],
        "binding:vif_type":"unbound",
        "device_owner":"",
        "binding:capabilities":{
           "port_filter":false
        },
        "mac_address":"fa:16:3e:80:14:5b",
        "fixed_ips":[

        ],
        "id":"6f9f6319-ce4b-4267-a5f8-558d6795632d",
        "security_groups":[
           "d30c3c54-5dba-49cf-a323-48a86f078d2d"
        ],
        "device_id":""
     },
     {
        "status":"ACTIVE",
        "binding:host_id":"devstack-havana",
        "name":"",
        "allowed_address_pairs":[
```

```
        ],
        "admin_state_up":true,
        "network_id":"d32019d3-bc6e-4319-9c1d-6722fc136a22",
        "tenant_id":"4fd44f30292945e481c7b8a0c8908869",
        "extra_dhcp_opts":[

        ],
        "binding:vif_type":"ovs",
        "device_owner":"network:dhcp",
        "binding:capabilities":{
           "port_filter":true
        },
        "mac_address":"fa:16:3e:3b:63:e8",
        "fixed_ips":[
           {
              "subnet_id":"54d6f61d-db07-451c-9ab3-b9609b6b6f0b",
              "ip_address":"192.0.0.2"
           }
        ],
        "id":"9842e9ab-7849-4bb5-8441-9fa223bfce45",
        "security_groups":[

        ],
        "device_id":"dhcp56016959-08ad-566c-8533-6240aff17dd5-d32019d3-
bc6e-4319-9c1d-6722fc136a22"
     },
     {
        "status":"ACTIVE",
        "binding:host_id":"devstack-havana",
        "name":"",
        "allowed_address_pairs":[

        ],
        "admin_state_up":true,
        "network_id":"db193ab3-96e3-4cb3-8fc5-05f4296d0324",
        "tenant_id":"26a7980765d0414dbc1fc1f88cdb7e6e",
        "extra_dhcp_opts":[

        ],
        "binding:vif_type":"ovs",
        "device_owner":"network:dhcp",
        "binding:capabilities":{
           "port_filter":true
        },
        "mac_address":"fa:16:3e:f7:80:62",
        "fixed_ips":[
           {
              "subnet_id":"08eae331-0402-425a-923c-34f7cfe39c1b",
              "ip_address":"10.0.0.3"
           }
        ],
        "id":"d2159251-552f-47ae-9960-f80d2aa6864f",
        "security_groups":[

        ],
        "device_id":"dhcp56016959-08ad-566c-8533-6240aff17dd5-
db193ab3-96e3-4cb3-8fc5-05f4296d0324"
     }
  ]
}
```

Example 3.43. Show port: XML response

```
<?xml version='1.0' encoding='UTF-8'?>
<ports xmlns="http://openstack.org/quantum/api/v2.0" xmlns:binding="http://
docs.openstack.org/ext/binding/api/v1.0" xmlns:quantum="http://openstack.org/
quantum/api/v2.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <port>
        <status>ACTIVE</status>
        <name />
        <admin_state_up quantum:type="bool">True</admin_state_up>
        <network_id>ebda9658-093b-41ba-80ce-0cf8cb8365d4</network_id>
        <tenant_id>63878e4c5dd649d2a980e37aefddfa87</tenant_id>
        <binding:vif_type>ovs</binding:vif_type>
        <device_owner>compute:None</device_owner>
        <binding:capabilities>
            <port_filter quantum:type="bool">False</port_filter>
        </binding:capabilities>
        <mac_address>fa:16:3e:b9:ef:05</mac_address>
        <fixed_ips>
            <fixed_ip>
                <subnet_id>aca4d43c-c48c-4a2c-9bb6-ba374ef7e135</subnet_id>
                <ip_address>172.24.4.227</ip_address>
            </fixed_ip>
        </fixed_ips>
        <id>664ebd1a-facd-4c20-948c-07a784475ab0</id>
        <device_id>f288bb5f-920d-4276-8345-2c0319c16f58</device_id>
    </port>
    <port>
        <status>DOWN</status>
        <name />
        <admin_state_up quantum:type="bool">True</admin_state_up>
        <network_id>ebda9658-093b-41ba-80ce-0cf8cb8365d4</network_id>
        <tenant_id />
        <binding:vif_type>ovs</binding:vif_type>
        <device_owner>network:router_gateway</device_owner>
        <binding:capabilities>
            <port_filter quantum:type="bool">False</port_filter>
        </binding:capabilities>
        <mac_address>fa:16:3e:4a:3a:a2</mac_address>
        <fixed_ips>
            <fixed_ip>
                <subnet_id>aca4d43c-c48c-4a2c-9bb6-ba374ef7e135</subnet_id>
                <ip_address>172.24.4.226</ip_address>
            </fixed_ip>
        </fixed_ips>
        <id>c5ca7017-c390-4ccc-8cd7-333747e57fef</id>
        <device_id>0dc517bf-9169-4aa6-88b7-569219962881</device_id>
    </port>
    <port>
        <status>ACTIVE</status>
        <name />
        <admin_state_up quantum:type="bool">True</admin_state_up>
        <network_id>9d83c053-b0a4-4682-ae80-c00df269ce0a</network_id>
        <tenant_id>625887121e364204873d362b553ab171</tenant_id>
        <binding:vif_type>ovs</binding:vif_type>
        <device_owner>network:router_interface</device_owner>
        <binding:capabilities>
            <port_filter quantum:type="bool">False</port_filter>
        </binding:capabilities>
        <mac_address>fa:16:3e:2d:dc:7e</mac_address>
```

```
        <fixed_ips>
            <fixed_ip>
                <subnet_id>a318fcb4-9ff0-4485-b78c-9e6738c21b26</subnet_id>
                <ip_address>10.0.0.1</ip_address>
            </fixed_ip>
        </fixed_ips>
        <id>d7815f5b-a228-47bb-a5e5-f139c4e476f6</id>
        <device_id>0dc517bf-9169-4aa6-88b7-569219962881</device_id>
    </port>
    <port>
        <status>ACTIVE</status>
        <name />
        <admin_state_up quantum:type="bool">True</admin_state_up>
        <network_id>9d83c053-b0a4-4682-ae80-c00df269ce0a</network_id>
        <tenant_id>625887121e364204873d362b553ab171</tenant_id>
        <binding:vif_type>ovs</binding:vif_type>
        <device_owner>network:dhcp</device_owner>
        <binding:capabilities>
            <port_filter quantum:type="bool">False</port_filter>
        </binding:capabilities>
        <mac_address>fa:16:3e:73:6d:1c</mac_address>
        <fixed_ips>
            <fixed_ip>
                <subnet_id>a318fcb4-9ff0-4485-b78c-9e6738c21b26</subnet_id>
                <ip_address>10.0.0.2</ip_address>
            </fixed_ip>
        </fixed_ips>
        <id>f8639521-fab2-4879-94b2-83a47bee8a26</id>
        <device_id>dhcpe1b8334f-9be9-5e49-aeee-b31e6df6c847-9d83c053-
b0a4-4682-ae80-c00df269ce0a</device_id>
    </port>
</ports>
```

This operation does not return a response body.

Update port

Method	URI	Description
PUT	/v2.0/ports/{port_id}	Updates a specified port.

You can update information for a port, such as its symbolic name and associated IPs. When you update IPs for a port, any previously associated IPs are removed, returned to the respective subnets allocation pools, and replaced by the IPs specified in the body for the update request. Therefore, this operation replaces the `fixed_ip` attribute when it is specified in the request body. If the updated IP addresses are not valid or are already in use, the operation fails and the existing IP addresses are not removed from the port.

When you update security groups for a port and the operation succeeds, any associated security groups are removed and replaced by the security groups specified in the body for the update request. Therefore, this operation replaces the `security_groups` attribute when you specify it in the request body. However, if the specified security groups are not valid, the operation fails and the existing security groups are not removed from the port.

Normal response codes: 200

Error response codes: badRequest (400), unauthorized (401), forbidden (403), itemNotFound (404), conflict (409)

Request

This table shows the URI parameters for the update port request:

Name	Type	Description
{port_id}	UUID	The UUID for the port of interest to you.

Example 3.44. Update port: JSON request

```
{
   "port":{
      "network_id":"ee2d3158-3e80-4fb3-ba87-c99f515d85e7",
      "admin_state_up":true
   }
}
```

Response

Example 3.45. Update port: JSON response

```
{
   "port":{
      "status":"DOWN",
      "binding:host_id":"",
      "name":"",
      "allowed_address_pairs":[

      ],
      "admin_state_up":true,
      "network_id":"ee2d3158-3e80-4fb3-ba87-c99f515d85e7",
```

```
      "tenant_id":"4fd44f30292945e481c7b8a0c8908869",
      "binding:vif_type":"unbound",
      "device_owner":"",
      "binding:capabilities":{
         "port_filter":false
      },
      "mac_address":"fa:16:3e:80:14:5b",
      "fixed_ips":[

      ],
      "id":"6f9f6319-ce4b-4267-a5f8-558d6795632d",
      "security_groups":[
         "d30c3c54-5dba-49cf-a323-48a86f078d2d"
      ],
      "device_id":""
   }
}
```

Example 3.46. Update port: XML response

```
<?xml version="1.0" encoding="UTF-8"?>
<port xmlns="http://openstack.org/quantum/api/v2.0"
    xmlns:binding="http://docs.openstack.org/ext/binding/api/v1.0"
    xmlns:quantum="http://openstack.org/quantum/api/v2.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <status>DOWN</status>
    <binding:host_id xsi:nil="true"/>
    <name>test_port_1</name>
    <admin_state_up quantum:type="bool">True</admin_state_up>
    <network_id>a3775a7d-9f8b-4148-be81-c84bbd0837ce</network_id>
    <tenant_id>60cd4f6dbc2f491982a284e7b83b5be3</tenant_id>
    <binding:vif_type>ovs</binding:vif_type>
    <device_owner/>
    <binding:capabilities>
        <port_filter quantum:type="bool">True</port_filter>
    </binding:capabilities>
    <mac_address>fa:16:3e:c9:8d:cf</mac_address>
    <fixed_ips quantum:type="list"/>
    <id>7f0aa3f1-883a-43b2-8d1b-e85fac52b417</id>
    <security_groups>
        <security_group>99f465bc-0d7c-4142-8829-7ae0179f2fa8</security_group>
    </security_groups>
    <device_id/>
</port>
```

This operation does not return a response body.

Delete port

Method	URI	Description
DELETE	`/v2.0/ports/{port_id}`	Deletes a specified port.

Any IP addresses that are associated with the port are returned to the respective subnets allocation pools.

Normal response codes: 204

Error response codes: unauthorized (401), forbidden (403), itemNotFound (404)

Request

This table shows the URI parameters for the delete port request:

Name	Type	Description
`{port_id}`	UUID	The UUID for the port of interest to you.

This operation does not require a request body.

4. API Extensions

The API extensions augment the core API. An API extension extends one or more of the following components of the core API:

- Resources. An extension defines new object classes.
- Attributes. An extended attribute defines a new attribute on existing resources. Prefixed by the extension name.
- Actions. An extended action defines a new operation on an existing resource.

Generic API extensions are not plug-in-specific. For information about plug-in-specific extensions that are shipped with OpenStack Networking, see the extension documentation in the source code tree.

List Available Extensions

You can list available extensions through the `/v2.0/extensions` URI.

Note

You must authenticate these requests just like any other API request.

The following example response shows a list of extensions:

```
Status Code: 200 OK
        Connection: keep-alive
        Content-Length: 654
        Content-Type: application/xml; charset=UTF-8
        Date: Wed, 12 Sep 2012 11:32:56 GMT
```

```
<?xml version="1.0" encoding="UTF-8"?>
<extensions>
    <extension>
```

```
        <updated>2012-07-20T10:00:00-00:00</updated>
        <name>Neutron L3 Router</name>
        <links/>
        <namespace>http://docs.openstack.org/ext/neutron/router/api/v1.0</
namespace>
        <alias>router</alias>
        <description>Router abstraction for basic L3 forwarding between
 L2 Neutron networks and access to external networks via a NAT gateway.</
description>
    </extension>
    <extension>
        <updated>2012-09-07T10:00:00-00:00</updated>
        <name>Provider Network</name>
        <links/>
        <namespace>http://docs.openstack.org/ext/provider/api/v1.0</namespace>
        <alias>provider</alias>
        <description>Expose mapping of virtual networks to physical networks</
description>
    </extension>
</extensions>
```

Also, you can query for specific extension using the extension alias. For instance, querying `/v2.0/extensions/router`, results in the following response:

```
Status Code: 200 OK
Connection: keep-alive
Content-Length: 350
Content-Type: application/xml; charset=UTF-8
Date: Wed, 12 Sep 2012 11:36:20 GMT
```

```
<?xml version="1.0" encoding="UTF-8"?>
<extension>
    <updated>2012-07-20T10:00:00-00:00</updated>
    <name>Neutron L3 Router</name>
    <links/>
    <namespace>http://docs.openstack.org/ext/neutron/router/api/v1.0</
namespace>
    <alias>router</alias>
    <description>Router abstraction for basic L3 forwarding between L2 Neutron
 networks and access to external networks via a NAT gateway.</description>
</extension>
```

The Provider Networks Extension (`provider`)

The provider networks extension allows OpenStack Networking API users with the appropriate rights, to specify how an OpenStack Networking network object is mapped to the underlying networking infrastructure. It also allows users with the appropriate rights to view such attributes when networks are queried.

To this aim, it extends the **network** resource by defining a set of attributes prefixed with the *provider* prefix, which specify these attributes.

Concepts

The provider networks extension is an attribute extension which adds the following set of attributes to the **network** resource:

- *provider:network_type* - Specifies the nature of the physical network mapped to this network resource. Examples are `flat`, `vlan`, or `gre`.
- *provider:physical_network* - Identifies the physical network on top of which this network object is being implemented. The OpenStack Networking API does not expose any facility for retrieving the list of available physical networks. As an example, in the Open vSwitch plug-in this is a symbolic name which is then mapped to specific bridges on each compute host through the Open vSwitch plug-in configuration file.
- *provider:segmentation_id* - Identifies an isolated segment on the physical network; the nature of the segment depends on the segmentation model defined by `network_type`. For instance, if `network_type` is `vlan`, then this is a `vlan` identifier; otherwise, if `network_type` is `gre`, then this will be a `gre` key.

The actual semantics of these attributes depend on the technology back end of the particular plug-in. See the plug-in documentation and the *OpenStack Cloud Administrator Guide* to understand which values should be specific for each of these attributes when OpenStack Networking is deployed with a particular plug-in. The examples shown in this chapter refer to the Open vSwitch plug-in.

It is also worth noting that the default policy settings allow only users with administrative rights to specify these parameters in requests, and to see their values in responses. By default, the provider network extension attributes are completely hidden from regular tenants. As a rule of thumb, if these attributes are not visible in a *GET /networks/<network-id>* operation, this implies the user submitting the request is not authorized to view or manipulate provider network attributes.

Network API operations with provider network extension

This section discusses operations for setting and retrieving the provider networks extension attributes for network objects.

List Networks

Verb	URI	Description
GET	/networks	Returns a list of networks with their provider networks attributes.

Normal Response Code: 200 OK

Error Response Codes: 401 Unauthorized

This operation returns, for each network, its provider network attributes as well as all the attributes normally returned by the list networks operation. Provider networks attribute are returned only if the user is authorized to view them.

Example 4.1. List Networks with provider attributes: JSON Response

```
{
  "networks": [
  {
    "status": "ACTIVE",
    "subnets": [],
    "name": "network-1",
    "admin_state_up": true,
    "tenant_id": "c1210485b2424d48804aad5d39c61b8f",
    "id": "3a06dfc7-d239-4aad-9a57-21cd171c72e5",
    "shared": false,
    "provider:network_type": "vlan",
    "provider:physical_network": "physnet_1",
    "provider:segmentation_id": 101
  },
  {
    "status": "ACTIVE",
    "subnets": [],
    "name": "network-2",
    "admin_state_up": true,
    "tenant_id": "c1210485b2424d48804aad5d39c61b8f",
    "id": "7db8c5a4-6eb0-478d-856b-7cfda2b25e13",
    "shared": false,
    "provider:network_type": "local",
    "provider:physical_network": null,
    "provider:segmentation_id": null
  }
 ]
}
```

Example 4.2. List Networks with provider attributes: XML Response

```
<?xml version='1.0' encoding='UTF-8'?>
<networks xmlns="http://openstack.org/neutron/api/v2.0"
    xmlns:provider="http://docs.openstack.org/ext/provider/api/v1.0"
    xmlns:neutron="http://openstack.org/neutron/api/v2.0"
```

```
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <network>
        <status>ACTIVE</status>
        <name>network-1</name>
        <admin_state_up neutron:type="bool">True</admin_state_up>
        <tenant_id>c1210485b2424d48804aad5d39c61b8f</tenant_id>
        <shared neutron:type="bool">False</shared>
        <id>3a06dfc7-d239-4aad-9a57-21cd171c72e5</id>
        <provider:network_type>vlan</provider:network_type>
        <provider:physical_network>physnet_1</provider:physical_network>
        <provider:segmentation_id neutron:type="int"
            >101</provider:segmentation_id>
    </network>
    <network>
        <status>ACTIVE</status>
        <name>network-2</name>
        <admin_state_up neutron:type="bool">True</admin_state_up>
        <tenant_id>c1210485b2424d48804aad5d39c61b8f</tenant_id>
        <shared neutron:type="bool">False</shared>
        <id>7db8c5a4-6eb0-478d-856b-7cfda2b25e13</id>
        <provider:network_type>local</provider:network_type>
        <provider:physical_network xsi:nil="true"/>
        <provider:segmentation_id xsi:nil="true"/>
    </network>
</networks>
```

Show Network

Verb	URI	Description
GET	/networks/*network_id*	Returns details about a specific network, including provider networks attributes.

Normal Response Code: 200 OK

Error Response Code: 401 Unauthorized, 404 Not Found

When the provider networks extension is enabled, and the user submitting the request is authorized to see provider networks mapping, this operation returns, for the network specified in the request URI, its provider network attributes, as well as all the attributes normally retuned by the show networks operation.

Example 4.3. Show network with provider attributes: JSON Response

```
{
  "network":
  {
    "status": "ACTIVE",
    "subnets": [],
    "name": "network-1",
    "admin_state_up": true,
    "tenant_id": "c1210485b2424d48804aad5d39c61b8f",
    "id": "3a06dfc7-d239-4aad-9a57-21cd171c72e5",
    "shared": false,
    "provider:network_type": "vlan",
    "provider:physical_network": "physnet_1",
    "provider:segmentation_id": 101
  }
```

```
}
```

Example 4.4. Show network with provider attributes: XML Response

```
<?xml version='1.0' encoding='UTF-8'?>
<network xmlns="http://openstack.org/neutron/api/v2.0"
    xmlns:provider="http://docs.openstack.org/ext/provider/api/v1.0"
    xmlns:neutron="http://openstack.org/neutron/api/v2.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <status>ACTIVE</status>
    <name>network-1</name>
    <admin_state_up neutron:type="bool">True</admin_state_up>
    <tenant_id>c1210485b2424d48804aad5d39c61b8f</tenant_id>
    <shared neutron:type="bool">False</shared>
    <id>3a06dfc7-d239-4aad-9a57-21cd171c72e5</id>
    <provider:network_type>vlan</provider:network_type>
    <provider:physical_network>physnet_1</provider:physical_network>
    <provider:segmentation_id neutron:type="int"
        >101</provider:segmentation_id>
</network>
```

Create Network

Verb	URI	Description
POST	/networks	Creates a new network and explicitly specify attributes with the underlying infrastructure using the provider network extension attributes.

Normal Response Code: 200 OK

Error Response Code: 400 Bad Request, 401 Unauthorized, 403 Forbidden

When the provider networks extension is enabled, and the user submitting the request is authorized to set provider networks mapping, this operation allows for specifying how a new network should be mapped on the underlying network infrastructure.

If the user submitting the request is not allowed to set provider networks attributes, a 403 Forbidden response will be returned.

As stated earlier in this chapter, the semantics of the various provider networks attribute vary with the particular plug-in employed. The following example shows how to create a network mapped to a specific vlan tag (the example refers to an OpenStack Networking deployment which uses the Open vSwitch plug-in).

Example 4.5. Create Network with provider attributes: JSON Request

```
{
 "network":
    {
      "name": "net-name",
      "admin_state_up": true,
      "provider:network_type": "vlan",
      "provider:physical_network": "physnet_1",
      "provider:segmentation_id": 201
    }
}
```

Example 4.6. Create Network with provider attributes: XML Request

```
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns="http://openstack.org/neutron/api/v2.0" xmlns:neutron="http://
openstack.org/neutron/api/v2.0" xmlns:provider="http://docs.openstack.org/ext/
provider/api/v1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <name>net-name</name>
    <admin_state_up neutron:type="bool">True</admin_state_up>
    <provider:network_type>vlan</provider:network_type>
    <provider:physical_network>physnet_1</provider:physical_network>
    <provider:segmentation_id neutron:type="int">201</
provider:segmentation_id>
</network>
```

Update Network

Verb	URI	Description
PUT	/networks/*network_id*	Updates a network, including its mapping with the underlying infrastructure using the provider network extension attributes.

Normal Response Code: 200 OK

Error Response Code: 400 Bad Request, 401 Unauthorized, 404 Not Found, 403 Forbidden

When the provider networks extension is enabled, and the user submitting the request is authorized to see provider networks mapping, this operation allows for specifying how an existing network should be mapped on the underlying network infrastructure.

If the user submitting the request is not allowed to set provider networks attributes, a 403 Forbidden response will be returned.

As stated earlier in this chapter, the semantics of the various provider networks attribute vary with the particular plug-in employed. The following example shows how to update a network in order to map it to a flat network (such as, no vlan tag); the example refers to an OpenStack Networking deployment that uses the Open vSwitch plug-in.

Example 4.7. Update provider attributes for a network: JSON Request

```
{
 "network":
    {
      "provider:network_type": "flat",
      "provider:physical_network": "physnet_1",
      "provider:segmentation_id": null
    }
}
```

Example 4.8. Update provider attributes for a network: XML Request

```
<?xml version='1.0' encoding='UTF-8'?>
<network xmlns="http://openstack.org/neutron/api/v2.0"
    xmlns:neutron="http://openstack.org/neutron/api/v2.0"
    xmlns:provider="http://docs.openstack.org/ext/provider/api/v1.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <provider:network_type>flat</provider:network_type>
    <provider:physical_network>physnet_1</provider:physical_network>
    <provider:segmentation_id xsi:nil="true"/>
</network>
```

The `binding` Extended Attributes for Ports

Use the Networking API v2.0 with the *`binding`* extended attributes to get information about, create, and update port objects.

The *`binding`*-prefixed extended attributes for ports are:

Table 4.1. *`binding`* Extended Attributes for Ports

Attribute	Type	Required	CRUD[a]	Default Value	Validation Constraints	Notes
`binding:vif_type`	String	N/A	R	None	N/A	Read-only. The vif type for the specified port.
`binding:host_id`	uuid-str	N/A	CRU	None	N/A	The ID of the host where the port is allocated. In some cases different implementations can run on different hosts.
`binding:profile`	list(dict)	N/A	CRU	None	N/A	A dictionary that enables the application running on the specified host to pass and receive vif port-specific information to the plug-in.
`binding:capabilities`	list(dict)	N/A	R	None	N/A	Read-only. A dictionary that enables the application to pass information about functions that Networking API v2.0 provides. Specify the following value: `port_filter : Boolean` to define whether Networking API v2.0 provides port filtering features such as security group and anti-MAC/ IP spoofing.
`binding:vnic_type`	String	N/A	CRU	normal	(normal, direct, macvtap)	The vnic type to be bound on the neutron port. To support SR-IOV PCI passthrough networking, you can request that the neutron port be realized as normal (virtual nic), direct (pci passthrough), or macvtap (virtual interface with a tap-like software interface). The ML2

Attribute	Type	Required	CRUD[a]	Default Value	Validation Constraints	Notes
						plug-in supports the vnic_type.

[a]

- **C**. Use the attribute in create operations.
- **R**. This attribute is returned in response to show and list operations.
- **U**. You can update the value of this attribute.
- **D**. You can delete the value of this attribute.

List Ports

Verb	URI	Description
GET	/ports	Lists ports to which the tenant has access. The *binding* extended attributes are visible to only administrative users. The *binding:vnic_type* extended attribute is visible to only port owners and administrative users.

Normal Response Code: 200

Error Response Codes: Unauthorized (401)

This operation lists ports to which the tenant has access.

This operation does not require a request body.

This operation returns a response body.

In addition to any other fields returned in the list port response, the following *binding*-prefixed fields are visible to administrative users:

Field	Description
binding:vif_type	Read-only. The vif type for the specified port.
binding:host_id	The ID of the host where the port is allocated. In some cases different implementations can run on different hosts.
binding:profile	A dictionary that enables the application running on the specified host to pass and receive vif port-specific information to the plug-in.
binding:capabilities	Read-only. A dictionary that enables the application to pass information about functions that Networking API v2.0 provides. Specify the following value: port_filter : Boolean to define whether Networking API v2.0 provides port filtering features such as security group and anti-MAC/IP spoofing.
binding:vnic_type	The vnic type to be bound on the neutron port. Valid values are normal, direct, or macvtap.

Show Port

Verb	URI	Description
GET	/ports/*port-id*	Shows information for a specified port. The *binding* extended attributes are visible to only administrative users. The *binding:vnic_type* extended attribute is visible to only port owners or administrative users.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Not Found (404)

This operation returns information for the port specified in the request URI.

This operation does not require a request body.

This operation returns a response body.

In addition to any fields returned in the show port response, the following *binding*-prefixed extended attributes are visible to administrative users:

Field	Description
binding:vif_type	Read-only. The vif type for the specified port.
binding:host_id	The ID of the host where the port is allocated. In some cases different implementations can run on different hosts.
binding:profile	A dictionary that enables the application running on the specified host to pass and receive vif port-specific information to the plug-in.
binding:capabilities	Read-only. A dictionary that enables the application to pass information about functions that Networking API v2.0 provides. Specify the following value: `port_filter : Boolean` to define whether Networking API v2.0 provides port filtering features such as security group and anti-MAC/IP spoofing.
binding:vnic_type	The requested vnic type to be bound on neutron port (can be either normal, direct or macvtap).

Create Port

Verb	URI	Description
POST	/ports	Creates a port on a specified network. Only administrative users can add the *binding* extended attributes.

Normal Response Code: 201

Error Response Codes: Bad Request (400), Unauthorized (401), Forbidden (403), Not Found (404), Conflict (409), MAC generation failure (503)

This operation creates an OpenStack Networking port. You must specify the network where the port is to created on the *network_id* attribute in the request body.

This operation requires a request body.

This operation returns a response body.

In addition to any attributes that can be set in a create port operation, administrative users can also set the following *binding*-prefixed extended attributes:

Field	Description
binding:host_id	The ID of the host where the port is allocated. In some cases different implementations can run on different hosts.
binding:profile	A dictionary that enables the application running on the specified host to pass and receive vif port-specific information to the plug-in.
binding:vnic_type	The vnic type to be bound on the neutron port. Valid values are normal, direct, or macvtap.

Update Port

Verb	URI	Description
PUT	/ports/*port-id*	Updates a specified port. Only administrative users can update the *binding* extended attributes. The *binding:vnic_type* extended attribute is visible to only port owners or administrative users.

Normal Response Code: 200

Error Response Codes: Bad Request (400), Unauthorized (401), Forbidden (403), Not Found (404), Conflict (409)

Use this operation to update information for a port.

This operation requires a request body.

This operation returns a response body.

In addition to any attributes that can be updated in the update port operation, administrative users can also update the following *binding*-prefixed extended attributes:

Field	Description
binding:host_id	The ID of the host where the port is allocated. In some cases different implementations can run on different hosts.
binding:profile	A dictionary that enables the application running on the specified host to pass and receive vif port-specific information to the plug-in.
binding:vnic_type	The vnic type to be bound on the neutron port. Valid values are normal, direct, or macvtap.

Layer-3 Networking Extension (router)

The Layer-3 networking extension enables OpenStack Networking API users to route packets between subnets, forward packets from internal networks to external ones, and access instances from external networks through *Floating IPs*.

This extension introduces the following resources:

- **router**, a logical entity for forwarding packets across internal subnets and NATting them on external networks through an appropriate external gateway.

- **floatingip**, representing an external IP address mapped to a OpenStack Networking port attached to an internal network.

Concepts

The OpenStack Networking layer-3 extension is a resource extension. It defines two new resources: **router** and **floatingip**.

A **router** is used to interconnect subnets and forward traffic among them. Another feature of the router is to NAT internal traffic to external networks.

A router has an interface for each subnet it is associated with. By default the IP address of such interface is the subnet's gateway IP. Also, whenever a router is associated with a subnet, a port for that router interface will be added to the subnet's network.

A **floating IP** is an IP address on an external network, which is associated with a specific port, and optionally a specific IP address, on a private OpenStack Networking network. Therefore a floating IP allows access to an instance on a private network from an external network. Floating IPs can only be defined on networks for which the attribute *router:external* (by the external network extension) has been set to True.

Table 4.2. Router Attributes

Attribute	Type	Required	CRUD[a]	Default Value	Validation Constraints	Notes
id	uuid-str	N/A	R	generated	N/A	Unique identifier for the router.
name	String	No	CRU	None	N/A	Human readable name for the router. Does not have to be unique
admin_state_up	Bool	No	CRU	true	{true | false }	Administrative state of the router
status	String	N/A	R	N/A	N/A	Indicates whether a router is currently operational or not
tenant_id	uuid-str	No	CR	Derived from Authentication token	N/A	Owner of the router. Only admin users can specify a tenant identifier other than its own
external_gateway_info	dict	No	CRU	None	No constraint	Information on external gateway for the router

[a]

- **C**. Use the attribute in create operations.
- **R**. This attribute is returned in response to show and list operations.
- **U**. You can update the value of this attribute.
- **D**. You can delete the value of this attribute.

Table 4.3. Floating IP Attributes

Attribute	Type	Required	CRUD	Default Value	Validation Constraints	Notes
id	uuid-str	N/A	R	generated	N/A	Unique identifier for the Floating IP instance
floating_network_id	uuid-str	Yes	CR	N/A	UUID Pattern	UUID of the external network where the floating IP is to be created
port_id	uuid-str	Yes	CRU	N/A	UUID Pattern	UUID of the port on an internal OpenStack Networking network which should be associated to the Floating IP
fixed_ip_address	IP Address	No	CRU	None	IP address or null	Specific IP address on `port_id` which should be associated with the floating IP
floating_ip_address	IP Address	N/A	R	Automatically allocated from pool	N/A	Address of the floating IP on the external network
tenant_id	uuid-str	No	CR	Derived from Authentication token	N/A	Owner of the Floating IP. Only admin users can specify a tenant identifier other than its own

Router API Operations

This section discusses operations for creating and managing routers through the OpenStack Networking L3 API extension.

List Routers

Verb	URI	Description
GET	/routers	Returns a list of logical routers accessible to the tenant submitting the request.

Normal Response Code: 200

Error Response Codes: Unauthorized (401)

This operation returns a list of routers to which the tenant has access. Default policy settings return only those routers that are owned by the tenant who submits the request, unless the request is submitted by an user with administrative rights. Users can control which attributes should be returned by using the *fields* query parameter. Additionally, results can be filtered by using query string parameters.

This operation does not require a request body; this operation returns a response body.

Example 4.9. List Routers: JSON Request

```
GET /v2.0/routers
Accept: application/json
```

Example 4.10. List Routers: JSON Response

```
{
   "routers":
   [{
       "status": "ACTIVE",
       "external_gateway_info": null,
       "name": "second_routers",
       "admin_state_up": true,
       "tenant_id": "6b96ff0cb17a4b859e1e575d221683d3",
       "id": "7177abc4-5ae9-4bb7-b0d4-89e94a4abf3b"
    },
    {
       "status": "ACTIVE",
       "external_gateway_info":
       {"network_id": "3c5bcddd-6af9-4e6b-9c3e-c153e521cab8"},
       "name": "router1",
       "admin_state_up": true,
       "tenant_id": "33a40233088643acb66ff6eb0ebea679",
       "id": "a9254bdb-2613-4a13-ac4c-adc581fba50d"
     }]
}
```

Show Router

Verb	URI	Description
GET	/routers/*router_id*	Returns details about a specific logical router.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Forbidden (403), Not Found (404)

This operation returns the details for a specific router whose identifier is specified on the request URI. Users can control which attributes should be returned by using the *fields* query parameter.

This operation does not require a request body.

This operation returns a response body.

Example 4.11. Show Router: JSON Request

```
GET /v2.0/routers/a9254bdb-2613-4a13-ac4c-adc581fba50d
Accept: application/json
```

Example 4.12. Show Router: JSON Response

```
{
   "router":
   {
       "status": "ACTIVE",
       "external_gateway_info":
       {
           "network_id": "3c5bcddd-6af9-4e6b-9c3e-c153e521cab8"
       },
       "name": "router1",
       "admin_state_up": true,
       "tenant_id": "33a40233088643acb66ff6eb0ebea679",
       "id": "a9254bdb-2613-4a13-ac4c-adc581fba50d"
     }
}
```

Create Router

Verb	URI	Description
POST	/routers	Create a new logical router.

Normal Response Code: 201

Error Response Codes: Unauthorized (401), Bad Request (400)

This operation creates a new logical router. When it is created, a logical router does not have any internal interface. In other words, it is not associated to any subnet. The user can optionally specify an external gateway for a router at create time; a router's external gateway must be plugged into an external network, that is to say a network for which the extended field `router:external` is set to true.

In order to specify an external gateway, the identifier of the external network must be passed in the request body's `external_gateway_info` attribute. This can be achieved as shown in the following snippet:

```
"external_gateway_info" :
{
    "network_id": <external_network_uuid>
}
```

This operation requires a request body.

This operation returns a response body.

Example 4.13. Create Router: JSON Request

```
{
   "router":{
      "name":"another_router",
      "admin_state_up":true
   }
}
```

Example 4.14. Create Router: JSON Response

```
{
    "router":
    {
        "status": "ACTIVE",
        "external_gateway_info": null,
        "name": "another_router",
        "admin_state_up": true,
        "tenant_id": "6b96ff0cb17a4b859e1e575d221683d3",
        "id": "8604a0de-7f6b-409a-a47c-a1cc7bc77b2e"
     }
}
```

Update Router

Verb	URI	Description
PUT	/routers/*router_id*	Updates a logical router.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Bad Request (400), Not Found (404)

This operation updates a logical router. Beyond the name and the administrative state, the only parameter which can be updated with this operation is the external gateway. For more information concerning how to set the external gateway for a router, please refer to the previous section (Create Router) or to the following example.

Please note that this operation does not allow to update router interfaces. To this aim, the add router interface and remove router interface should be used.

Example 4.15. Update Router: JSON Request

```
{
   "router":{
      "external_gateway_info":{
         "network_id":"8ca37218-28ff-41cb-9b10-039601ea7e6b"
      }
   }
}
```

Example 4.16. Update Router: JSON Response

```
{
    "router":
    {
        "status": "ACTIVE",
        "external_gateway_info":
        {
            "network_id": "8ca37218-28ff-41cb-9b10-039601ea7e6b"
        },
        "name": "another_router",
        "admin_state_up": true,
        "tenant_id": "6b96ff0cb17a4b859e1e575d221683d3",
        "id": "8604a0de-7f6b-409a-a47c-a1cc7bc77b2e"
    }
}
```

Delete Router

Verb	URI	Description
DELETE	/routers/*router_id*	Removes a logical router and its external gateway interface, if it exists

Normal Response Code: 204

Error Response Codes: Unauthorized (401), Not Found (404), Conflict (409)

This operation removes a logical router; the operation will fail if the router still has some internal interfaces.

Users must remove all router interfaces before deleting the router, by removing all internal interfaces through remove router interface operation.

This operation does not require a request body.

This operation does not return a response body.

Example 4.17. Delete Router: JSON Request

```
DELETE /v2.0/routers/8604a0de-7f6b-409a-a47c-a1cc7bc77b2e
Accept: application/json
```

Add Interface to Router

Verb	URI	Description
PUT	/routers/*router_id*/ add_router_interface	Adds an internal interface to a logical router.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Bad Request (400), Not Found (404), Conflict (409)

This operation attaches a subnet to an internal router interface. Either a subnet identifier or a port identifier must be passed in the request body. If both are specified, a `400 Bad Request` error is returned.

When the `subnet_id` attribute is specified in the request body, the subnet's gateway ip address is used to create the router interface; otherwise, if `port_id` is specified, the IP address associated with the port is used for creating the router interface.

It is worth remarking that a `400 Bad Request` error is returned if several IP addresses are associated with the specified port, or if no IP address is associated with the port; also a `409 Conflict` is returned if the port is already used.

Example 4.18. Add Router Interface: JSON Request

```
PUT /v2.0/routers/8604a0de-7f6b-409a-a47c-a1cc7bc77b2e/add_router_interface
Accept: application/json

{"subnet_id": "a2f1f29d-571b-4533-907f-5803ab96ead1"}
```

Example 4.19. Add Router Interface: JSON Response

```
{
    "subnet_id": "a2f1f29d-571b-4533-907f-5803ab96ead1",
    "port_id": "3a44f4e5-1694-493a-a1fb-393881c673a4"
}
```

The port identifier returned by this operation can either be the same identifier passed in the request body or the identifier of a new port created by this operation to attach the subnet specified by `subnet_id` to the router.

It is also worth noting that after executing this operation, the `device_id` attribute of this port is set to the router's identifier, and the `device_owner` attribute is set to `network:router_interface`, as shown in the following example:

```
{
    "port":
    {
        "status": "ACTIVE",
        "name": "",
        "admin_state_up": true,
        "network_id": "5307648b-e836-4658-8f1a-ff7536870c64",
        "tenant_id": "6b96ff0cb17a4b859e1e575d221683d3",
        "device_owner": "network:router_interface",
        "mac_address": "fa:16:3e:f7:d1:9c",
        "fixed_ips":
        [{
            "subnet_id": "a2f1f29d-571b-4533-907f-5803ab96ead1",
            "ip_address": "10.1.1.1"
        }],
        "id": "3a44f4e5-1694-493a-a1fb-393881c673a4",
        "device_id": "7177abc4-5ae9-4bb7-b0d4-89e94a4abf3b"
    }
}
```

Remove Interface from Router

Verb	URI	Description
PUT	/routers/*router_id*/remove_router_interface	Removes an internal interface from a logical router.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Bad Request (400), Not Found (404), Conflict (409)

This request requires a request body.

This request returns a response body.

This operation removes an internal router interface, thus detaching a subnet from the router.

This operation removes an internal router interface, which detaches a subnet from the router. You must specify either a subnet ID or port ID in the request body; this value is used to identify the router interface to remove.

You can also specify both a subnet ID and port ID. If you specify both IDs, the subnet ID must correspond to the subnet ID of the first IP address on the port specified by the port ID. Otherwise, the operation returns a 409 Conflict error. The response contains information about the affected router and interface.

A 404 Not Found error will be returned either if the router or the subnet/port do not exist or are not visible to the user.

As a consequence of this operation, the port connecting the router with the subnet is removed from the subnet's network.

Example 4.20. Remove Router Interface: JSON Request

```
PUT /v2.0/routers/8604a0de-7f6b-409a-a47c-a1cc7bc77b2e/remove_router_interface
Accept: application/json

{"subnet_id": "a2f1f29d-571b-4533-907f-5803ab96ead1"}
```

Example 4.21. Remove Router Interface: Response

```
{
    "id": "8604a0de-7f6b-409a-a47c-a1cc7bc77b2e",
    "tenant_id": "2f245a7b-796b-4f26-9cf9-9e82d248fda7",
    "port_id": "3a44f4e5-1694-493a-a1fb-393881c673a4",
    "subnet_id": "a2f1f29d-571b-4533-907f-5803ab96ead1"
}
```

Floating IP API Operations

This section discusses operations for creating and managing floating IPs through the OpenStack Networking L3 API extension.

List Floating IPs

Verb	URI	Description
GET	/floatingips	Returns a list of floating IPs accessible to the tenant submitting the request.

Normal Response Code: 200 OK

Error Response Codes: 401 Unauthorized

This operation does not require a request body.

This operation returns a response body.

The operation returns a list of floating IPs to which the tenant has access. Default policy settings return only those floating IPs that are owned by the tenant who submits the request, unless the request is submitted by an user with administrative rights. Users can control which attributes should be returned by using the *fields* query parameter. Additionally, results can be filtered by using query string parameters.

Example 4.22. List Floating IPs: JSON Request

```
GET /v2.0/floatingips
Accept: application/json
```

Example 4.23. List Floating IPs: JSON Response

```
{
 "floatingips":
    [
        {
         "router_id": "d23abc8d-2991-4a55-ba98-2aaea84cc72f",
         "tenant_id": "4969c491a3c74ee4af974e6d800c62de",
         "floating_network_id": "376da547-b977-4cfe-9cba-275c80debf57",
         "fixed_ip_address": "10.0.0.3",
         "floating_ip_address": "172.24.4.228",
         "port_id": "ce705c24-c1ef-408a-bda3-7bbd946164ab",
         "id": "2f245a7b-796b-4f26-9cf9-9e82d248fda7"
        },
        {
         "router_id": null,
         "tenant_id": "4969c491a3c74ee4af974e6d800c62de",
         "floating_network_id": "376da547-b977-4cfe-9cba-275c80debf57",
         "fixed_ip_address": null,
         "floating_ip_address": "172.24.4.227",
         "port_id": null,
         "id": "61cea855-49cb-4846-997d-801b70c71bdd"
        }
    ]
}
```

Show Floating IP

Verb	URI	Description
GET	/floatingips/*floatingip_id*	Returns details about a specific Floating IP.

Normal Response Code: 200 OK

Error Response Code: 401 Unauthorized, 404 Not Found

This operation does not require a request body.

This operation returns a response body.

This operation returns the details for a specific floating IP whose identifier is specified on the request URI. Users can control which attributes should be returned by using the *fields* query parameter.

Example 4.24. Show Floating IP: JSON Request

```
GET /v2.0/floatingips/2f245a7b-796b-4f26-9cf9-9e82d248fda7.json?
fields=fixed_ip_address&fields=floating_ip_address
Accept: application/json
```

Example 4.25. Show Floating IP: JSON Response

```
{
 "floatingip":
    {
     "fixed_ip_address": "10.0.0.3",
     "floating_ip_address": "172.24.4.228"
    }
}
```

Create Floating IP

Verb	URI	Description
POST	/floatingips	Creates a floating IP, and configures its association with an internal port, if the relevant information are specified

Normal Response Code: 200 OK

Error Response Code: 400 Bad Request, 401 Unauthorized, 409 Conflict

This operation requires a request body.

This operation returns a response body.

This operation creates a floating IP. Users can specify association information for this floating IP in the request body. These information are however optional, and can be specified with an UPDATE call at a later stage. The only mandatory parameter in the request body is the external network identifier, `floating_network_id`. Floating IPs can only be created on external networks. If the network specified by `floating_network_id` is not external (for example, router:external=False), a 400 error is returned.

An address for the floating ip will be automatically allocated, unless the `floating_ip_address` attribute is specified in the request body. If the requested floating IP address does not fall in the external network's subnet range, a 400 error will be returned. If the requested floating IP address is already in use, a 409 error code will be returned.

Users can associate the floating IP with an internal port using the `port_id` attribute in the request body. If an invalid port identifier is specified, a 404 error will be returned. The internal OpenStack Networking port associated with the Floating IP must have at least an IP address configured, otherwise a 400 error will be returned.

As an OpenStack Networking port might be associated with multiple IP addresses, the particular IP address to associate with the floating IP can be specified using the `fixed_ip_address` request body parameter. The default logic of this operation is to associate the floating IP with a single IP address configured on a port; hence, if a port has multiple IP addresses, it is mandatory to specify the `fixed_ip_address` attribute. If an invalid IP address is specified in `fixed_ip_address` a 400 error will be returned.

If the internal OpenStack Networking port and ip address selected for association are already associated to another floating IP, a 409 error will be returned.

Example 4.26. Create Floating IP: JSON Request

```
POST /v2.0/floatingips
Accept: application/json

{
    "floatingip":
    {
        "floating_network_id": "376da547-b977-4cfe-9cba-275c80debf57",
        "port_id": "ce705c24-c1ef-408a-bda3-7bbd946164ab"
     }
}
```

Example 4.27. Create Floating IP: JSON Response

```
{
    "floatingip":
    {
        "router_id": "d23abc8d-2991-4a55-ba98-2aaea84cc72f",
        "tenant_id": "4969c491a3c74ee4af974e6d800c62de",
        "floating_network_id": "376da547-b977-4cfe-9cba-275c80debf57",
        "fixed_ip_address": "10.0.0.3",
        "floating_ip_address": "172.24.4.228",
        "port_id": "ce705c24-c1ef-408a-bda3-7bbd946164ab",
        "id": "2f245a7b-796b-4f26-9cf9-9e82d248fda7"
     }
}
```

Update Floating IP

Verb	URI	Description
PUT	/floatingips/*floatingip_id*	Updates a floating IP and its association with an internal port, if the relevant information are specified in the request body.

Normal Response Code: 200 OK

Error Response Code: 400 Bad Request, 401 Unauthorized, 404 Not Found, 409 Conflict

This operation requires a request body.

This operation returns a response body.

This operation updates an association between a floating IP and an OpenStack Networking port. The association process is exactly the same as the one discussed for the *create floating IP* operation.

To disassociate a floating IP from a port, set the `port_id` request attribute to null or omit it from the request body.

Example 4.28. Update Floating IP (associate): JSON Request

```
{
   "floatingip":{
      "port_id":"fc861431-0e6c-4842-a0ed-e2363f9bc3a8"
   }
}
```

Example 4.29. Update Floating IP (associate): JSON Response

```
{
    "floatingip":
    {
        "router_id": "d23abc8d-2991-4a55-ba98-2aaea84cc72f",
        "tenant_id": "4969c491a3c74ee4af974e6d800c62de",
        "floating_network_id": "376da547-b977-4cfe-9cba-275c80debf57",
        "fixed_ip_address": "10.0.0.4",
        "floating_ip_address": "172.24.4.228",
        "port_id": "fc861431-0e6c-4842-a0ed-e2363f9bc3a8",
        "id": "2f245a7b-796b-4f26-9cf9-9e82d248fda7"
     }
 }
```

Example 4.30. Update Floating IP (disassociate): JSON Request

```
{
   "floatingip":{
      "port_id":null
   }
}
```

Example 4.31. Update Floating IP (disassociate): JSON Response

```
{
    "floatingip":
    {
        "router_id": "d23abc8d-2991-4a55-ba98-2aaea84cc72f",
        "tenant_id": "4969c491a3c74ee4af974e6d800c62de",
        "floating_network_id": "376da547-b977-4cfe-9cba-275c80debf57",
        "fixed_ip_address": null,
        "floating_ip_address": "172.24.4.228",
        "port_id": null,
        "id": "2f245a7b-796b-4f26-9cf9-9e82d248fda7"
     }
 }
```

Delete Floating IP

Verb	URI	Description
DELETE	/ floatingips/ *floatingip_id*	Removes a floating IP and its association information.

Normal Response Code: 204

Error Response Code: 401 Unauthorized, 404 Not Found

This operation does not require a request body.

This operation does not return a response body.

The operation removes the floating IP whose identifier is specified in the request URI. If the floating IP being removed is associated with an OpenStack Networking port, the association is removed as well.

Example 4.32. Delete Floating IP: JSON Request

```
DELETE /v2.0/floatingips/2f245a7b-796b-4f26-9cf9-9e82d248fda7
Accept: application/json
```

Configurable external gateway modes extension

By default, when a gateway is attached to a router using the Neutron L3 extension, Network Address Translation (NAT) is enabled for traffic generated by subnets attached to the router. With this extension, the user will have the option of choosing whether SNAT should be enabled or not on a router basis.

This is achieved simply by specifying a boolean attribute, `enable_snat`, in the `external_gateway_info` attribute of the `router` resource.

This extension redefines the external_gateway_info attribute as depicted in the following table.

Table 4.4. external_gateway_info attributes

Attribute	Type	Required	Default Value	Validation Constraints	Notes
network_id	UUID	Yes	N/A	Must be a valid uuid	

Attribute	Type	Required	Default Value	Validation Constraints	Notes
				representative of an external network.	
enable_snat	Boolean	No	True	{True|False}	The default setting is `True` to ensure backward compatibility for plugins supporting this extension.

SNAT can be enabled or disabled at any time on a Neutron router regardless of the current status of floating IPs. Floating IPs will continue working even when SNAT is disabled.

List Routers

Verb	URI	Description
GET	/routers	Retrieve Neutron routers

Success and error response codes are not changed with regards to the operation as introduced by the L3 API extension.

When this extension is enabled, this operation will also return the current Source NAT status for configured routers, as shown in the sample below.

The response for the *show router* operation is the same, with the obvious exception that a single router is returned.

Example 4.33. Router list with configurable external gateway modes enabled

```
{
   "routers":
   [{
       "status": "ACTIVE",
       "external_gateway_info":
       {"network_id": "3c5bcddd-6af9-4e6b-9c3e-c153e521cab8",
        "enable_snat": true},
       "name": "second_router",
       "admin_state_up": true,
       "tenant_id": "6b96ff0cb17a4b859e1e575d221683d3",
       "id": "7177abc4-5ae9-4bb7-b0d4-89e94a4abf3b"
    },
    {
       "status": "ACTIVE",
       "external_gateway_info":
       {"network_id": "3c5bcddd-6af9-4e6b-9c3e-c153e521cab8",
        "enable_snat": false},
       "name": "router1",
       "admin_state_up": true,
       "tenant_id": "33a40233088643acb66ff6eb0ebea679",
       "id": "a9254bdb-2613-4a13-ac4c-adc581fba50d"
     }]
}
```

Create Router with external gateway

Verb	URI	Description
POST	/routers	Create a new Neutron router

Success and error response codes are not changed with regards to the operation as introduced by the L3 API extension.

Neutron API users can specify whether SNAT should be performed on the network specified as the router's external gateway by setting `enable_snat` in `external_gateway_info` to either `True` or `False`; the default value is `True`.

Example 4.34. Create router with SNAT disabled

```
POST /v2.0/routers
Accept: application/json
{
    "router":
    {
        "name": "another_router",
        "admin_state_up": true,
        "external_gateway_info": {
            "network_id": "3c5bcddd-6af9-4e6b-9c3e-c153e521cab8",
            "enable_snat": false}
    }
}

HTTP/1.1 201 OK
Content-Type: application/json; charset=UTF-8
{
    "router":
    {
        "status": "ACTIVE",
        "external_gateway_info": {
            "network_id": "3c5bcddd-6af9-4e6b-9c3e-c153e521cab8",
            "enable_snat": false}
        "name": "another_router",
        "admin_state_up": true,
        "tenant_id": "6b96ff0cb17a4b859e1e575d221683d3",
        "id": "8604a0de-7f6b-409a-a47c-a1cc7bc77b2e"
     }
}
```

Update Router's external gateway info

Verb	URI	Description
PUT	/routers/`router_id`	Create a new Neutron router

Success and error response codes are not changed with regards to the operation as introduced by the L3 API extension.

Neutron API users can enable or disable SNAT on a router specifying the `enable_snat` attribute in the `external_gateway_info` attribute for the router resource. This operation can be either used for updating the SNAT status only, the external network, or both attributes at the same time. In any case, if the `enable_snat` attribute is not specified, it will default to `True`. For instance, if the current SNAT status is disabled, and the router's gateway is updated to a different external network without specifying `enable_snat`, SNAT will be enabled for the new network.

It is important to note that whenever updating a router's external gateway information, the `network_id`parameter must be specified always, even if the final goal is just to enable or disable SNAT for the router on the same external network.

The rest of this section provides some samples for updating a router's external gateway info with SNAT mode.

Example 4.35. Disable SNAT for the current external network

```
{
    "router":
    {
        "name": "another_router",
        "admin_state_up": true,
        "external_gateway_info": {
            "network_id": "3c5bcddd-6af9-4e6b-9c3e-c153e521cab8"}
    }
}
```

```
{
   "router":{
      "status":"ACTIVE",
      "external_gateway_info":{
         "network_id":"3c5bcddd-6af9-4e6b-9c3e-c153e521cab8",
         "enable_snat":true
      },
      "name":"another_router",
      "admin_state_up":true,
      "tenant_id":"6b96ff0cb17a4b859e1e575d221683d3",
      "id":"8604a0de-7f6b-409a-a47c-a1cc7bc77b2e"
   }
}
```

```
{
   "router":{
      "status":"ACTIVE",
      "external_gateway_info":{
         "network_id":"3c5bcddd-6af9-4e6b-9c3e-c153e521cab8",
         "enable_snat":false
      },
      "name":"another_router",
      "admin_state_up":true,
      "tenant_id":"6b96ff0cb17a4b859e1e575d221683d3",
      "id":"8604a0de-7f6b-409a-a47c-a1cc7bc77b2e"
   }
}
```

```
{
   "router":{
      "external_gateway_info":{
         "network_id":"3c5bcddd-6af9-4e6b-9c3e-c153e521cab8",
         "enable_snat":false
      }
   }
}
```

Example 4.36. Change external network and enable SNAT

```
{
    "router":
    {
        "name": "another_router",
        "admin_state_up": true,
        "external_gateway_info": {
            "network_id": "3c5bcddd-6af9-4e6b-9c3e-c153e521cab8",
            "enable_snat": false}
    }
}
```

```
{
    "router":
    {
        "status": "ACTIVE",
        "external_gateway_info": {
            "network_id": "3c5bcddd-6af9-4e6b-9c3e-c153e521cab8",
            "enable_snat": false},
        "name": "another_router",
        "admin_state_up": true,
        "tenant_id": "6b96ff0cb17a4b859e1e575d221683d3",
        "id": "8604a0de-7f6b-409a-a47c-a1cc7bc77b2e"
     }
}
```

```
{
    "router":
    {
        "external_gateway_info": {
            "network_id": "002ab3b9-9127-4158-be30-4b45f3814df5"}
    }
}
```

```
{
    "router":
    {
        "status": "ACTIVE",
        "external_gateway_info": {
            "network_id": "002ab3b9-9127-4158-be30-4b45f3814df5",
            "enable_snat": true},
        "name": "another_router",
        "admin_state_up": true,
        "tenant_id": "6b96ff0cb17a4b859e1e575d221683d3",
        "id": "8604a0de-7f6b-409a-a47c-a1cc7bc77b2e"
     }
}
```

Example 4.37. Change external network and keep SNAT disabled

```
{
    "router":
    {
        "name": "another_router",
        "admin_state_up": true,
        "external_gateway_info": {
            "network_id": "3c5bcddd-6af9-4e6b-9c3e-c153e521cab8",
            "enable_snat": false}
    }
}
```

```
{
    "router":
    {
        "status": "ACTIVE",
        "external_gateway_info": {
            "network_id": "3c5bcddd-6af9-4e6b-9c3e-c153e521cab8",
            "enable_snat": false},
        "name": "another_router",
        "admin_state_up": true,
        "tenant_id": "6b96ff0cb17a4b859e1e575d221683d3",
        "id": "8604a0de-7f6b-409a-a47c-a1cc7bc77b2e"
     }
}
```

```
{
    "router":
    {
        "external_gateway_info": {
            "network_id": "002ab3b9-9127-4158-be30-4b45f3814df5",
            "enable_snat": false}
    }
}
```

```
{
    "router":
    {
        "status": "ACTIVE",
        "external_gateway_info": {
            "network_id": "002ab3b9-9127-4158-be30-4b45f3814df5",
            "enable_snat": false},
        "name": "another_router",
        "admin_state_up": true,
        "tenant_id": "6b96ff0cb17a4b859e1e575d221683d3",
        "id": "8604a0de-7f6b-409a-a47c-a1cc7bc77b2e"
     }
}
```

Quotas

The `neutron.conf` configuration file defines default quota values that are applied to all tenants. This extension enables an administrative user to define quotas values on a per-tenant basis. For example, an administrative user can permit tenant A to create at most X networks and tenant B to create at most Y networks.

Method	URI	Description
GET	`/v2.0/quotas`	Lists quotas for tenants who have non-default quota values.
GET	`/v2.0/quotas/{tenant_id}`	Shows quotas for a specified tenant.
PUT	`/v2.0/quotas/{tenant_id}`	Updates quotas for a specified tenant. Use when non-default quotas are desired.
DELETE	`/v2.0/quotas/{tenant_id}`	Resets quotas to default values for a specified tenant.

List quotas

Method	URI	Description
GET	`/v2.0/quotas`	Lists quotas for tenants who have non-default quota values.

Normal response codes: 200

Error response codes: unauthorized (401), forbidden (403)

Request

This operation does not require a request body.

Response

Example 4.38. List quotas: JSON response

```
{
    "quotas": [{
        "subnet": 10,
        "network": 10,
        "floatingip": 50,
        "tenant_id": "b7445f221cda4f4a8ac7db6b218b1339",
        "router": 10,
        "port": 30
    }]
}
```

Show quota

Method	URI	Description
GET	/v2.0/quotas/{tenant_id}	Shows quotas for a specified tenant.

Normal response codes: 200

Error response codes: unauthorized (401), forbidden (403)

Request

This table shows the URI parameters for the show quota request:

Name	Type	Description
{tenant_id}	UUID	The tenant ID in a multi-tenancy cloud.

This operation does not require a request body.

Response

Example 4.39. Show quota: JSON response

```
{
    "quota": {
        "subnet": 10,
        "router": 10,
        "port": 50,
        "network": 10,
        "floatingip": 50
    }
}
```

Update quota

Method	URI	Description
PUT	/v2.0/quotas/{tenant_id}	Updates quotas for a specified tenant. Use when non-default quotas are desired.

Normal response codes: 200

Error response codes: unauthorized (401), forbidden (403)

Request

This table shows the URI parameters for the update quota request:

Name	Type	Description
{tenant_id}	UUID	The tenant ID in a multi-tenancy cloud.

Example 4.40. Update quota: JSON request

```
{
    "quota": {
        "subnet": 40,
        "router": 50,
        "network": 10,
        "floatingip": 30,
        "port": 30
    }
}
```

Response

Example 4.41. Update quota: JSON response

```
{
    "quota": {
        "subnet": 40,
        "router": 50,
        "port": 30,
        "network": 10,
        "floatingip": 30
    }
}
```

Reset quota

Method	URI	Description
DELETE	`/v2.0/quotas/{tenant_id}`	Resets quotas to default values for a specified tenant.

Normal response codes: 204

Error response codes: unauthorized (401), forbidden (403)

Request

This table shows the URI parameters for the reset quota request:

Name	Type	Description
`{tenant_id}`	UUID	The tenant ID in a multi-tenancy cloud.

This operation does not require a request body.

Security groups and rules (security-groups)

Method	URI	Description
GET	`/v2.0/security-groups`	Lists all OpenStack Networking security groups to which the specified tenant has access.
POST	`/v2.0/security-groups`	Creates an OpenStack Networking security group.
GET	`/v2.0/security-groups/{security_group_id}`	Shows information for a specified security group.
DELETE	`/v2.0/security-groups/{security_group_id}`	Deletes an OpenStack Networking security group.
GET	`/v2.0/security-group-rules`	Lists a summary of all OpenStack Networking security group rules that the specified tenant can access.
POST	`/v2.0/security-group-rules`	Creates an OpenStack Networking security group rule.
GET	`/v2.0/security-group-rules/{rules-security-groups-id}`	Shows detailed information for a specified security group rule.
DELETE	`/v2.0/security-group-rules/{rules-security-groups-id}`	Deletes a specified rule from a OpenStack Networking security group.

List security groups

Method	URI	Description
GET	/v2.0/security-groups	Lists all OpenStack Networking security groups to which the specified tenant has access.

The list shows the unique ID for each security group and the rules that are associated with each security group.

Normal response codes: 200

Error response codes: unauthorized (401)

Request

Example 4.42. List security groups: JSON request

```
GET /v2.0/security-groups
Accept: application/json
```

This operation does not require a request body.

Response

Example 4.43. List security groups: JSON response

```
{
   "security_groups":[
      {
         "description":"default",
         "id":"85cc3048-abc3-43cc-89b3-377341426ac5",
         "name":"default",
         "security_group_rules":[
            {
               "direction":"egress",
               "ethertype":"IPv6",
               "id":"3c0e45ff-adaf-4124-b083-bf390e5482ff",
               "port_range_max":null,
               "port_range_min":null,
               "protocol":null,
               "remote_group_id":null,
               "remote_ip_prefix":null,
               "security_group_id":"85cc3048-abc3-43cc-89b3-377341426ac5",
               "tenant_id":"e4f50856753b4dc6afee5fa6b9b6c550"
            },
            {
               "direction":"egress",
               "ethertype":"IPv4",
               "id":"93aa42e5-80db-4581-9391-3a608bd0e448",
               "port_range_max":null,
               "port_range_min":null,
               "protocol":null,
               "remote_group_id":null,
               "remote_ip_prefix":null,
               "security_group_id":"85cc3048-abc3-43cc-89b3-377341426ac5",
```

```
                "tenant_id":"e4f50856753b4dc6afee5fa6b9b6c550"
            },
            {
                "direction":"ingress",
                "ethertype":"IPv6",
                "id":"c0b09f00-1d49-4e64-a0a7-8a186d928138",
                "port_range_max":null,
                "port_range_min":null,
                "protocol":null,
                "remote_group_id":"85cc3048-abc3-43cc-89b3-377341426ac5",
                "remote_ip_prefix":null,
                "security_group_id":"85cc3048-abc3-43cc-89b3-377341426ac5",
                "tenant_id":"e4f50856753b4dc6afee5fa6b9b6c550"
            },
            {
                "direction":"ingress",
                "ethertype":"IPv4",
                "id":"f7d45c89-008e-4bab-88ad-d6811724c51c",
                "port_range_max":null,
                "port_range_min":null,
                "protocol":null,
                "remote_group_id":"85cc3048-abc3-43cc-89b3-377341426ac5",
                "remote_ip_prefix":null,
                "security_group_id":"85cc3048-abc3-43cc-89b3-377341426ac5",
                "tenant_id":"e4f50856753b4dc6afee5fa6b9b6c550"
            }
        ],
        "tenant_id":"e4f50856753b4dc6afee5fa6b9b6c550"
    }
  ]
}
```

Create security group

Method	URI	Description
POST	/v2.0/security-groups	Creates an OpenStack Networking security group.

This operation creates a security group with default security group rules for the IPv4 and IPv6 ether types.

Normal response codes: 201

Error response codes: badRequest (400), unauthorized (401)

Request

Example 4.44. Create security group: JSON request

```
{
   "security_group":{
      "name":"new-webservers",
      "description":"security group for webservers"
   }
}
```

Response

Example 4.45. Create security group: JSON response

```
{
   "security_group":{
      "description":"security group for webservers",
      "id":"2076db17-a522-4506-91de-c6dd8e837028",
      "name":"new-webservers",
      "security_group_rules":[
         {
            "direction":"egress",
            "ethertype":"IPv4",
            "id":"38ce2d8e-e8f1-48bd-83c2-d33cb9f50c3d",
            "port_range_max":null,
            "port_range_min":null,
            "protocol":null,
            "remote_group_id":null,
            "remote_ip_prefix":null,
            "security_group_id":"2076db17-a522-4506-91de-c6dd8e837028",
            "tenant_id":"e4f50856753b4dc6afee5fa6b9b6c550"
         },
         {
            "direction":"egress",
            "ethertype":"IPv6",
            "id":"565b9502-12de-4ffd-91e9-68885cff6ae1",
            "port_range_max":null,
            "port_range_min":null,
            "protocol":null,
            "remote_group_id":null,
            "remote_ip_prefix":null,
            "security_group_id":"2076db17-a522-4506-91de-c6dd8e837028",
```

```
            "tenant_id":"e4f50856753b4dc6afee5fa6b9b6c550"
         }
      ],
      "tenant_id":"e4f50856753b4dc6afee5fa6b9b6c550"
   }
}
```

Show security group

Method	URI	Description
GET	/v2.0/security-groups/{security_group_id}	Shows information for a specified security group.

This operation returns a response body that contains the description, name, ID, and security group rules associated with the specified security group and tenant ID.

Normal response codes: 200

Error response codes: unauthorized (401), itemNotFound (404)

Request

This table shows the URI parameters for the show security group request:

Name	Type	Description
{security_group_id}	Uuid	The unique identifier of the security group.

Example 4.46. Show security group: JSON request

```
GET /v2.0/security-groups/85cc3048-abc3-43cc-89b3-377341426ac5
Accept: application/json
```

This operation does not require a request body.

Response

Example 4.47. Show security group: JSON response

```
{
   "security_group":{
      "description":"default",
      "id":"85cc3048-abc3-43cc-89b3-377341426ac5",
      "name":"default",
      "security_group_rules":[
         {
            "direction":"egress",
            "ethertype":"IPv6",
            "id":"3c0e45ff-adaf-4124-b083-bf390e5482ff",
            "port_range_max":null,
            "port_range_min":null,
            "protocol":null,
            "remote_group_id":null,
            "remote_ip_prefix":null,
            "security_group_id":"85cc3048-abc3-43cc-89b3-377341426ac5",
            "tenant_id":"e4f50856753b4dc6afee5fa6b9b6c550"
         },
         {
            "direction":"egress",
            "ethertype":"IPv4",
            "id":"93aa42e5-80db-4581-9391-3a608bd0e448",
            "port_range_max":null,
```

```
            "port_range_min":null,
            "protocol":null,
            "remote_group_id":null,
            "remote_ip_prefix":null,
            "security_group_id":"85cc3048-abc3-43cc-89b3-377341426ac5",
            "tenant_id":"e4f50856753b4dc6afee5fa6b9b6c550"
         },
         {
            "direction":"ingress",
            "ethertype":"IPv6",
            "id":"c0b09f00-1d49-4e64-a0a7-8a186d928138",
            "port_range_max":null,
            "port_range_min":null,
            "protocol":null,
            "remote_group_id":"85cc3048-abc3-43cc-89b3-377341426ac5",
            "remote_ip_prefix":null,
            "security_group_id":"85cc3048-abc3-43cc-89b3-377341426ac5",
            "tenant_id":"e4f50856753b4dc6afee5fa6b9b6c550"
         },
         {
            "direction":"ingress",
            "ethertype":"IPv4",
            "id":"f7d45c89-008e-4bab-88ad-d6811724c51c",
            "port_range_max":null,
            "port_range_min":null,
            "protocol":null,
            "remote_group_id":"85cc3048-abc3-43cc-89b3-377341426ac5",
            "remote_ip_prefix":null,
            "security_group_id":"85cc3048-abc3-43cc-89b3-377341426ac5",
            "tenant_id":"e4f50856753b4dc6afee5fa6b9b6c550"
         }
      ],
      "tenant_id":"e4f50856753b4dc6afee5fa6b9b6c550"
   }
}
```

Delete security group

Method	URI	Description
DELETE	`/v2.0/security-groups/{security_group_id}`	Deletes an OpenStack Networking security group.

This operation deletes an OpenStack Networking security group and its associated security group rules, provided that a port is not associated with the security group.

This operation does not require a request body. This operation does not return a response body.

Normal response codes: 204

Error response codes: unauthorized (401), itemNotFound (404)

Request

This table shows the URI parameters for the delete security group request:

Name	Type	Description
`{security_group_id}`	Uuid	The unique identifier of the security group.

Example 4.48. Delete security group: JSON request

```
DELETE /v2.0/security-groups/e470bdfc-4869-459b-a561-cb3377efae59
Content-Type: application/json
Accept: application/json
```

This operation does not require a request body.

Response

Example 4.49. Delete security group: JSON response

```
status: 204
```

This operation does not return a response body.

List security group rules

Method	URI	Description
GET	`/v2.0/security-group-rules`	Lists a summary of all OpenStack Networking security group rules that the specified tenant can access.

The list provides the unique ID for each security group rule.

Normal response codes: 200

Error response codes: unauthorized (401)

Request

Example 4.50. List security group rules: JSON request

```
GET /v2.0/security-group-rules/
Accept: application/json
```

This operation does not require a request body.

Response

Example 4.51. List security group rules: JSON response

```
{
   "security_group_rules":[
      {
         "direction":"egress",
         "ethertype":"IPv6",
         "id":"3c0e45ff-adaf-4124-b083-bf390e5482ff",
         "port_range_max":null,
         "port_range_min":null,
         "protocol":null,
         "remote_group_id":null,
         "remote_ip_prefix":null,
         "security_group_id":"85cc3048-abc3-43cc-89b3-377341426ac5",
         "tenant_id":"e4f50856753b4dc6afee5fa6b9b6c550"
      },
      {
         "direction":"egress",
         "ethertype":"IPv4",
         "id":"93aa42e5-80db-4581-9391-3a608bd0e448",
         "port_range_max":null,
         "port_range_min":null,
         "protocol":null,
         "remote_group_id":null,
         "remote_ip_prefix":null,
         "security_group_id":"85cc3048-abc3-43cc-89b3-377341426ac5",
         "tenant_id":"e4f50856753b4dc6afee5fa6b9b6c550"
      },
      {
         "direction":"ingress",
         "ethertype":"IPv6",
         "id":"c0b09f00-1d49-4e64-a0a7-8a186d928138",
```

```
         "port_range_max":null,
         "port_range_min":null,
         "protocol":null,
         "remote_group_id":"85cc3048-abc3-43cc-89b3-377341426ac5",
         "remote_ip_prefix":null,
         "security_group_id":"85cc3048-abc3-43cc-89b3-377341426ac5",
         "tenant_id":"e4f50856753b4dc6afee5fa6b9b6c550"
      },
      {
         "direction":"ingress",
         "ethertype":"IPv4",
         "id":"f7d45c89-008e-4bab-88ad-d6811724c51c",
         "port_range_max":null,
         "port_range_min":null,
         "protocol":null,
         "remote_group_id":"85cc3048-abc3-43cc-89b3-377341426ac5",
         "remote_ip_prefix":null,
         "security_group_id":"85cc3048-abc3-43cc-89b3-377341426ac5",
         "tenant_id":"e4f50856753b4dc6afee5fa6b9b6c550"
      }
   ]
}
```

Create security group rule

Method	URI	Description
POST	/v2.0/security-group-rules	Creates an OpenStack Networking security group rule.

Normal response codes: 201

Error response codes: badRequest (400), unauthorized (401), itemNotFound (404), buildInProgress (409)

Request

Example 4.52. Create security group rule: JSON request

```
{
   "security_group_rule":{
      "direction":"ingress",
      "port_range_min":"80",
      "ethertype":"IPv4",
      "port_range_max":"80",
      "protocol":"tcp",
      "remote_group_id":"85cc3048-abc3-43cc-89b3-377341426ac5",
      "security_group_id":"a7734e61-b545-452d-a3cd-0189cbd9747a"
   }
}
```

Response

Example 4.53. Create security group rule: JSON response

```
{
   "security_group_rule":{
      "direction":"ingress",
      "ethertype":"IPv4",
      "id":"2bc0accf-312e-429a-956e-e4407625eb62",
      "port_range_max":80,
      "port_range_min":80,
      "protocol":"tcp",
      "remote_group_id":"85cc3048-abc3-43cc-89b3-377341426ac5",
      "remote_ip_prefix":null,
      "security_group_id":"a7734e61-b545-452d-a3cd-0189cbd9747a",
      "tenant_id":"e4f50856753b4dc6afee5fa6b9b6c550"
   }
}
```

Show security group rule

Method	URI	Description
GET	/v2.0/security-group-rules/{rules-security-groups-id}	Shows detailed information for a specified security group rule.

The response body contains the following information about the security group rule:

Normal response codes: 200

Error response codes: unauthorized (401), itemNotFound (404)

Request

This table shows the URI parameters for the show security group rule request:

Name	Type	Description
{rules-security-groups-id}	Uuid	The unique identifier of the security group rule.

Example 4.54. Show security group rule: JSON request

```
GET /v2.0/security-group-rules/ 3c0e45ff-adaf-4124-b083-bf390e5482ff
Accept: application/json
```

This operation does not require a request body.

Response

Example 4.55. Show security group rule: JSON response

```
{
   "security_group_rule":{
      "direction":"egress",
      "ethertype":"IPv6",
      "id":"3c0e45ff-adaf-4124-b083-bf390e5482ff",
      "port_range_max":null,
      "port_range_min":null,
      "protocol":null,
      "remote_group_id":null,
      "remote_ip_prefix":null,
      "security_group_id":"85cc3048-abc3-43cc-89b3-377341426ac5",
      "tenant_id":"e4f50856753b4dc6afee5fa6b9b6c550"
   }
}
```

Delete security group rule

Method	URI	Description
DELETE	`/v2.0/security-group-rules/{rules-security-groups-id}`	Deletes a specified rule from a OpenStack Networking security group.

Normal response codes: 204

Error response codes: unauthorized (401), itemNotFound (404)

Request

This table shows the URI parameters for the delete security group rule request:

Name	Type	Description
`{rules-security-groups-id}`	Uuid	The unique identifier of the security group rule.

Example 4.56. Delete security group rule: JSON request

```
DELETE /v2.0/security-group-rules/fc3c327a-b5b5-4cd3-9577-52893289ce08
Content-Type: application/json
Accept: application/json
```

This operation does not require a request body.

Response

Example 4.57. Delete security group rule: JSON response

```
status: 204
```

This operation does not return a response body.

The Agent Management Extension

In a typical OpenStack Networking deployment, some agents run on network or compute nodes, such as `neutron-dhcp-agent`, `neutron-ovs-agent`, and `neutron-l3-agent`. This extension provides a way for administrators(enforced by the policy engine) to view their status and update their attributes. Updating agent management API attributes will affect operations of other components such as OpenStack Networking schedulers. For instance administrators can disable a certain agent, and OpenStack Networking schedulers will not schedule resources to it.

For how to use agent management extension and OpenStack Networking schedulers feature, see the *OpenStack Cloud Administrator Guide*.

Verb	URI	Description
GET	/agents	List agents which report their status to OpenStack Networking server.
GET	/agents/*agent_id*	Show information of the given agent.
PUT	/agents/*agent_id*	Update the agent's admin status and description.

Verb	URI	Description
DELETE	/agents/*agent_id*	Delete the given agent.

List Agents

Verb	URI	Description
GET	/agents	List agents which report their status to OpenStack Networking server.

Normal Response Code: 200

This operation does not require a request body.

This operation returns a response body. The default policy behavior is that non-admin user won't be able to see any agent in the response when this call is invoked

Example 4.58. List Agents: JSON Request

```
GET /v2.0/agents HTTP/1.1
Host: controlnode:9696
User-Agent: python-neutronclient
Content-Type: application/json
Accept: application/json
X-Auth-Token: c52a1b304fec4ca0ac85dc1741eec6e2
```

Example 4.59. List Agents: JSON Response

```
{
   "agents":[
      {
         "binary":"neutron-dhcp-agent",
         "description":null,
         "admin_state_up":false,
         "heartbeat_timestamp":"2013-03-26T09:35:13.000000",
         "alive":false,
         "id":"af4567ad-c2e6-4311-944d-22efc12f89af",
         "topic":"dhcp_agent",
         "host":"HostC",
         "agent_type":"DHCP agent",
         "started_at":"2013-03-26T09:35:01.000000",
         "created_at":"2013-03-26T09:35:01.000000",
         "configurations":{
            "subnets":2,
            "use_namespaces":true,
            "dhcp_driver":"neutron.agent.linux.dhcp.Dnsmasq",
            "networks":2,
            "dhcp_lease_time":120,
            "ports":3
         }
      }
   ]
}
```

Show Agent

Verb	URI	Description
GET	/agents/*agent_id*	Show information of the given agent.

Normal Response Code: 200

Error Response Codes:NotFound (404) if not authorized or the agent does not exist

This operation returns information for the given agent.

This operation does not require a request body.

This operation returns a response body. The body contents depend on the agent's type.

Example 4.60. Show Agent: JSON Request

```
GET /v2.0/agents/af4567ad-c2e6-4311-944d-22efc12f89af HTTP/1.1
Host: controlnode:9696
User-Agent: python-neutronclient
Content-Type: application/json
Accept: application/json
X-Auth-Token: a54d6fdda41341f892150e2aaf648f0d
```

Example 4.61. Show Agent: JSON Response

```
{
   "agent":{
      "binary":"neutron-dhcp-agent",
      "description":null,
      "admin_state_up":false,
      "heartbeat_timestamp":"2013-03-26T09:35:13.000000",
      "alive":false,
      "id":"af4567ad-c2e6-4311-944d-22efc12f89af",
      "topic":"dhcp_agent",
      "host":"HostC",
      "agent_type":"DHCP agent",
      "started_at":"2013-03-26T09:35:01.000000",
      "created_at":"2013-03-26T09:35:01.000000",
      "configurations":{
         "subnets":2,
         "use_namespaces":true,
         "dhcp_driver":"neutron.agent.linux.dhcp.Dnsmasq",
         "networks":2,
         "dhcp_lease_time":120,
         "ports":3
      }
   }
}
```

Update Agent

Verb	URI	Description
PUT	/agents/*agent_id*	Update the agent's admin status and description.

Normal Response Code: 200

Error Response Codes: BadRequest (400) if something other than description or admin status is changed, NotFound (404) if not authorized or the agent does not exist

This operation updates the agent's admin status and description.

This operation requires a request body.

This operation returns a response body.

Example 4.62. Update Agent: JSON Request

```
PUT /v2.0/agents/af4567ad-c2e6-4311-944d-22efc12f89af HTTP/1.1
Host: controlnode:9696
User-Agent: python-neutronclient
Content-Type: application/json
Accept: application/json
X-Auth-Token: 4cbb09e780434b249ff596d6979fd8fc
Content-Length: 38

{
    "agent": {
        "admin_state_up": "False"
    }
}
```

Example 4.63. Update Agents: JSON Response

```
{
   "agent":{
      "binary":"neutron-dhcp-agent",
      "description":null,
      "admin_state_up":false,
      "heartbeat_timestamp":"2013-03-26T09:35:13.000000",
      "alive":false,
      "id":"af4567ad-c2e6-4311-944d-22efc12f89af",
      "topic":"dhcp_agent",
      "host":"HostC",
      "agent_type":"DHCP agent",
      "started_at":"2013-03-26T09:35:01.000000",
      "created_at":"2013-03-26T09:35:01.000000",
      "configurations":{
         "subnets":2,
         "use_namespaces":true,
         "dhcp_driver":"neutron.agent.linux.dhcp.Dnsmasq",
         "networks":2,
         "dhcp_lease_time":120,
         "ports":3
      }
   }
```

```
}
```

Delete Agent

Verb	URI	Description
DELETE	/agents/*agent_id*	Delete the given agent.

Normal Response Code: 204

Error Response Codes: NotFound (404) if not authorized or the agent does not exist

This operation deletes the agent.

This operation does not require a request body.

This operation does not return a response body.

Example 4.64. Delete Agent: JSON Request

```
DELETE /v2.0/agents/44002aeb-2817-4cb8-9306-34308b4b40d9 HTTP/1.1
Host: controlnode:9696
User-Agent: python-neutronclient
Content-Type: application/json
Accept: application/json
X-Auth-Token: 4cbb09e780434b249ff596d6979fd8fc
```

Example 4.65. Delete Agent: JSON Response

```
HTTP/1.1 204 No Content
Content-Length: 0
Date: Tue, 26 Mar 2013 12:12:35 GMT
```

The ExtraRoute Extension

You can set up route configuration on the Router using this extension. This extension augments the 'Router' resource by adding a new 'routes' attribute.

You can specify a set of nexthop IPs and destination CIDRs. Note the nexthop IP must be a part of one of the subnets router interfaces are connected to. This is why configuration of route property is allowed only update operation on REST.

Table 4.5. Router Attributes

Attribute	Type	Required	CRUD[a]	Default Value	Validation Constraints	Notes
routes	list of dict	No	U	None	List should be in this form. [{'nexthop':IPAddress, 'destination':CIDR}]	Extra route configuration

[a]

- **C**. Use the attribute in create operations.
- **R**. This attribute is returned in response to show and list operations.
- **U**. You can update the value of this attribute.
- **D**. You can delete the value of this attribute.

Update Extra route

Verb	URI	Description
PUT	/routers/*router_id*	Update logical router with routes attribute.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Bad Request (400), Not Found (404), Conflict (409)

This operation configures extra routes on the router. The nexthop IP must be a part of one of the subnets to which the router interfaces are connected. Otherwise, the server responds with `400 Bad Request`. When a validation error is detected, such as a format error of IP address or CIDR, the server responds with `400 Bad Request`. When Networking receives a request to delete the router interface for subnets that are used by one or more routes, it responds with `409 Conflict`.

Example 4.66. Update the routes attribute for a given router

```
{
   "router":{
      "routes":[
         {
            "nexthop":"10.1.0.10",
            "destination":"40.0.1.0/24"
         }
      ]
   }
}
```

Example 4.67. Update routes: Response

```
{"router":
    {"status": "ACTIVE",
     "external_gateway_info": {"network_id": "5c26e0bb-
a9a9-429c-9703-5c417a221096"},
     "name": "router1",
     "admin_state_up": true,
     "tenant_id": "936fa220b2c24a87af51026439af7a3e",
     "routes": [{"nexthop": "10.1.0.10", "destination": "40.0.1.0/24"}],
     "id": "babc8173-46f6-4b6f-8b95-38c1683a4e22"}
}
```

The Load Balancer as a Service (LBaaS) extension

The LBaaS extension enables OpenStack tenants to load-balance their VM traffic. The extension enables you to:

- Load-balance client traffic from one network to application services, such as VMs, on the same or a different network.
- Load-balance several protocols, such as TCP and HTTP.
- Monitor the health of application services.
- Support session persistence.

Concepts

This extension introduces these concepts:

VIP	The primary load balancing configuration object. Specifies the virtual IP address and port where client traffic is received. Also defines other details such as the load balancing method to be used, protocol, and so on. This entity is sometimes known in load-balancing products as a *virtual server*, *vserver*, or *listener*.
pool	A logical set of devices, such as web servers, that you group together to receive and process traffic. The load balancing function chooses which member of the pool handles the new requests or connections received on the VIP address. Each VIP has one pool.
pool member	The application that runs on the back-end server.
health monitor	Determines whether or not back-end members of the VIP pool can process a request. A pool can have several health monitors associated with it. The LBaaS extension supports these types of health monitors: • **PING**. Pings the members by using ICMP. • **TCP**. Connects to the members by using TCP. • **HTTP**. Sends an HTTP request to the member. • **HTTPS**. Sends a secure HTTP request to the member. When a pool has several monitors associated with it, all monitors check each member of the pool. If any monitor declares a member as unhealthy, the member status is

	changed to **inactive** and the member does not participate in the load balancing for the pool. All monitors must declare the member to be healthy for it to stay **active**.
session persistence	Forces connections or requests in the same session to be processed by the same member as long as it is active. The LBaaS extension supports these types of persistence: • **SOURCE_IP**. All connections that originate from the same source IP address are handled by the same member of the pool. • **HTTP_COOKIE**. The load balancing function creates a cookie on the first request from a client. Subsequent requests that contain the same cookie value are handled by the same member of the pool. • **APP_COOKIE**. The load balancing function relies on a cookie established by the back-end application. All requests with the same cookie value are handled by the same member of the pool. Absence of `session_persistence` attribute means no session persistence mechanism is used. When no session persistence is used, the `session_persistence` attribute does not appear in the API response, instead of returning null. You can clear session persistence for the VIP by sending `null` in `session_persistence` attribute in a VIP update request.

High-level task flow

To use the LBaaS extension to configure load balancing, you must complete these high-level tasks:

1. Create a pool, which is initially empty.
2. Create one or several members in the pool.
3. Create one or several health monitors.
4. Associate the health monitors with the pool.
5. Create a VIP that is associated with the pool.

VIP operations

This section discusses operations for managing load balancer VIPs through the LBaaS extension.

Table 4.6. VIP attributes

Attribute	Type	Required	CRUD [a]	Default value	Validation constraints	Notes
id	uuid-str	N/A	R	generated	N/A	Unique identifier for the VIP.
tenant_id	uuid-str	Yes	CR	Derived from authentication token.	N/A	Owner of the VIP. Only an admin user can specify a tenant identifier other than its own.
name	String	No	CRU	None	N/A	Human readable name for the VIP. Does not have to be unique.
description	String	No	CRU	None	N/A	Human readable description for the VIP.
subnet_id	uuid-str	No	CR	None	UUID pattern.	The subnet on which to allocate the VIP address.
address	IP	No	CR	None	IP address or null	The IP address of the VIP.
protocol	String	Yes	CR	None	{ "TCP" | "HTTP" | "HTTPS" }	The protocol of the VIP address.
protocol_port	Integer	Yes	CR	None	[0..65535]	The port on which to listen to client traffic that is associated with the VIP address.
pool_id	uuid-str	No	CRU	None	UUID pattern.	The `pool` that the VIP associated with.
session_persistence	dict	No	CRU	None	composite [b]	Session persistence parameters of the VIP. Absence of `session_persistence` attribute means no session persistence mechanism is used. When no session persistence is used, the `session_persistence` attribute does not appear in the API response, instead of returning null. You can clear session persistence for the VIP by sending `null` in `session_persistence` attribute in a VIP update request.
connection_limit	Integer	No	CRU	-1	None	The maximum number of connections allowed for the VIP or `-1` if the limit is not set.
admin_state_up	Bool	No	CRU	true	{true | false }	Administrative state of the VIP
status	String	N/A	R	N/A	N/A	Indicates whether a VIP is currently operational.

[a]• **C**. Use the attribute in create operations.

• **R**. This attribute is returned in response to show and list operations.

- **U**. You can update the value of this attribute.
- **D**. You can delete the value of this attribute.

[b]Session persistence is a dictionary with the following attributes:

- **type** - any of `APP_COOKIE`, `HTTP_COOKIE` or `SOURCE_IP`
- **cookie_name** - any string, required if type is `APP_COOKIE`

List VIPs

Verb	URI	Description
GET	/lb/vips	Lists VIPs.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Forbidden (403)

This operation does not require a request body.

This operation returns a response body.

Example 4.68. List VIPs: HTTP request

```
GET /v2.0/lb/vips.json HTTP/1.1
User-Agent: python-neutronclient
Accept: application/json
```

Example 4.69. List VIPs: JSON response

```
{
   "vips":[
      {
         "status":"ACTIVE",
         "protocol":"HTTP",
         "description":"",
         "admin_state_up":true,
         "subnet_id":"8032909d-47a1-4715-90af-5153ffe39861",
         "tenant_id":"83657cfcdfe44cd5920adaf26c48ceea",
         "connection_limit":1000,
         "pool_id":"72741b06-df4d-4715-b142-276b6bce75ab",
         "session_persistence":{
            "cookie_name":"MyAppCookie",
            "type":"APP_COOKIE"
         },
         "address":"10.0.0.10",
         "protocol_port":80,
         "port_id":"b5a743d6-056b-468b-862d-fb13a9aa694e",
         "id":"4ec89087-d057-4e2c-911f-60a3b47ee304",
         "name":"my-vip"
      }
   ]
}
```

Show VIP details

Verb	URI	Description
GET	/lb/vips/*vip-id*	Shows details for a specified VIP.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Forbidden (403), Not Found (404)

This operation does not require a request body.

This operation returns a response body.

Example 4.70. Show VIP details: HTTP request

```
GET /v2.0/lb/vips/4ec89087-d057-4e2c-911f-60a3b47ee304.json HTTP/1.1
User-Agent: python-neutronclient
Accept: application/json
```

Example 4.71. Show VIP details: JSON response

```
{
   "vip":{
      "status":"ACTIVE",
      "protocol":"HTTP",
      "description":"",
      "admin_state_up":true,
      "subnet_id":"8032909d-47a1-4715-90af-5153ffe39861",
      "tenant_id":"83657cfcdfe44cd5920adaf26c48ceea",
      "connection_limit":1000,
      "pool_id":"72741b06-df4d-4715-b142-276b6bce75ab",
      "session_persistence":{
         "cookie_name":"MyAppCookie",
         "type":"APP_COOKIE"
      },
      "address":"10.0.0.10",
      "protocol_port":80,
      "port_id":"b5a743d6-056b-468b-862d-fb13a9aa694e",
      "id":"4ec89087-d057-4e2c-911f-60a3b47ee304",
      "name":"my-vip"
   }
}
```

Create VIP

Verb	URI	Description
POST	/lb/vips	Creates a load balancer VIP.

Normal Response Code: 201

Error Response Codes: Unauthorized (401), Bad Request (400)

This operation requires a request body.

This operation returns a response body.

Example 4.72. Create VIP: HTTP and JSON request

```
POST /v2.0/lb/vips.json HTTP/1.1
User-Agent: python-neutronclient
Accept: application/json
```

```
{
   "vip":{
      "protocol":"HTTP",
      "name":"NewVip",
      "admin_state_up":true,
      "subnet_id":"8032909d-47a1-4715-90af-5153ffe39861",
      "pool_id":"61b1f87a-7a21-4ad3-9dda-7f81d249944f",
      "protocol_port":"80"
   }
}
```

Example 4.73. Create VIP: HTTP and JSON response

```
HTTP/1.1 201 Created
Content-Type: application/json; charset=UTF-8
```

```
{
   "vip":{
      "status":"PENDING_CREATE",
      "protocol":"HTTP",
      "description":"",
      "admin_state_up":true,
      "subnet_id":"8032909d-47a1-4715-90af-5153ffe39861",
      "tenant_id":"83657cfcdfe44cd5920adaf26c48ceea",
      "connection_limit":-1,
      "pool_id":"61b1f87a-7a21-4ad3-9dda-7f81d249944f",
      "address":"10.0.0.11",
      "protocol_port":80,
      "port_id":"f7e6fe6a-b8b5-43a8-8215-73456b32e0f5",
      "id":"c987d2be-9a3c-4ac9-a046-e8716b1350e2",
      "name":"NewVip"
   }
}
```

Update VIP

Verb	URI	Description
PUT	/lb/vips/*vip-id*	Updates a load balancer VIP.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Bad Request (400), Not Found (404)

Example 4.74. Update VIP: HTTP and JSON request

```
PUT /v2.0/lb/vips/c987d2be-9a3c-4ac9-a046-e8716b1350e2.json
User-Agent: python-neutronclient
Accept: application/json
```

```
{
   "vip":{
      "connection_limit":"1000"
   }
}
```

Example 4.75. Update VIP: HTTP and JSON response

```
HTTP/1.1 200 OK
```

```
Content-Type: application/json; charset=UTF-8
```

```
{
   "vip":{
      "status":"PENDING_UPDATE",
      "protocol":"HTTP",
      "description":"",
      "admin_state_up":true,
      "subnet_id":"8032909d-47a1-4715-90af-5153ffe39861",
      "tenant_id":"83657cfcdfe44cd5920adaf26c48ceea",
      "connection_limit":1000,
      "pool_id":"61b1f87a-7a21-4ad3-9dda-7f81d249944f",
      "address":"10.0.0.11",
      "protocol_port":80,
      "port_id":"f7e6fe6a-b8b5-43a8-8215-73456b32e0f5",
      "id":"c987d2be-9a3c-4ac9-a046-e8716b1350e2",
      "name":"NewVip"
   }
}
```

Delete VIP

Verb	URI	Description
DELETE	/lb/vips/*vip-id*	Removes a load balancer VIP.

Normal Response Code: 204

Error Response Codes: Unauthorized (401), Not Found (404), Conflict (409)

This operation does not require a request body.

This operation does not return a response body.

Example 4.76. Delete VIP: HTTP request

```
DELETE /v2.0/lb/vips/c987d2be-9a3c-4ac9-a046-e8716b1350e2.json HTTP/1.1
Accept: application/json
```

Example 4.77. Delete VIP: HTTP response

```
HTTP/1.1 204 No Content
Content-Length: 0
```

Pool operations

This section discusses operations for managing load balancer pools through the Load balancing as a service extension.

Table 4.7. Pool Attributes

Attribute	Type	Required	CRUD [a]	Default Value	Validation Constraints	Notes
id	uuid-str	N/A	R	generated	N/A	Unique identifier for the pool.
tenant_id	uuid-str	Yes	CR	Derived from authentication token.	N/A	Owner of the pool. Only an admin user can specify a

Attribute	Type	Required	CRUD [a]	Default Value	Validation Constraints	Notes
						tenant identifier other than its own
vip_id	uuid-str	No	R	None	UUID pattern.	The `vip` that the pool associated with.
name	String	No	CRU	None	N/A	Human readable name for the pool. Does not have to be unique.
description	String	No	CRU	None	N/A	Human readable description for the pool.
subnet_id	uuid-str	No	CR	None	UUID pattern.	The network that pool members belong to.
protocol	String	Yes	CR	None	{ "TCP" | "HTTP" | "HTTPS" }	The protocol of the pool.
lb_method	String	Yes	CRU	None	None	The algorithm used to distribute load between the members of the pool.
health_monitors	uuid-list	No	CRU	None	N/A	List of health monitors to associate with the pool.
members	uuid-list	No	R	None	N/A	List of members that belong to the pool.
admin_state_up	Bool	No	CRU	true	{true | false }	Administrative state of the pool.
status	String	N/A	R	N/A	N/A	Indicates whether a pool is currently operational or not.

[a]
- **C**. Use the attribute in create operations.
- **R**. This attribute is returned in response to show and list operations.
- **U**. You can update the value of this attribute.
- **D**. You can delete the value of this attribute.

List pools

Verb	URI	Description
GET	/lb/pools	Lists pools.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Forbidden (403)

This operation does not require a request body.

This operation returns a response body.

Example 4.78. List pools: HTTP request

```
GET /v2.0/lb/pools.json HTTP/1.1
User-Agent: python-neutronclient
Accept: application/json
```

Example 4.79. List pools: HTTP and JSON response

```
HTTP/1.1 200 OK
```

```
Content-Type: application/json; charset=UTF-8
```


```
{
   "pools":[
      {
         "status":"ACTIVE",
         "lb_method":"ROUND_ROBIN",
         "protocol":"HTTP",
         "description":"",
         "health_monitors":[
            "466c8345-28d8-4f84-a246-e04380b0461d",
            "5d4b5228-33b0-4e60-b225-9b727c1a20e7"
         ],
         "subnet_id":"8032909d-47a1-4715-90af-5153ffe39861",
         "tenant_id":"83657cfcdfe44cd5920adaf26c48ceea",
         "admin_state_up":true,
         "name":"app_pool",
         "members":[
            "701b531b-111a-4f21-ad85-4795b7b12af6",
            "beb53b4d-230b-4abd-8118-575b8fa006ef"
         ],
         "id":"72741b06-df4d-4715-b142-276b6bce75ab",
         "vip_id":"4ec89087-d057-4e2c-911f-60a3b47ee304"
      }
   ]
}
```


Show pool

Verb	URI	Description
GET	/lb/pools/*pool-id*	Returns details about a specific pool.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Forbidden (403), Not Found (404)

This operation does not require a request body.

This operation returns a response body.

Example 4.80. Show pool: HTTP request

```
GET /v2.0/lb/pools/72741b06-df4d-4715-b142-276b6bce75ab.json HTTP/1.1
User-Agent: python-neutronclient
Accept: application/json
```

Example 4.81. Show pool: HTTP and JSON response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
```


```
{
   "pool":{
      "status":"ACTIVE",
      "lb_method":"ROUND_ROBIN",
      "protocol":"HTTP",
```

```
      "description":"",
      "health_monitors":[
         "466c8345-28d8-4f84-a246-e04380b0461d",
         "5d4b5228-33b0-4e60-b225-9b727c1a20e7"
      ],
      "subnet_id":"8032909d-47a1-4715-90af-5153ffe39861",
      "tenant_id":"83657cfcdfe44cd5920adaf26c48ceea",
      "admin_state_up":true,
      "name":"app_pool",
      "members":[
         "701b531b-111a-4f21-ad85-4795b7b12af6",
         "beb53b4d-230b-4abd-8118-575b8fa006ef"
      ],
      "id":"72741b06-df4d-4715-b142-276b6bce75ab",
      "vip_id":"4ec89087-d057-4e2c-911f-60a3b47ee304"
   }
}
```

Create Pool

Verb	URI	Description
POST	/lb/pools	Create a new load balancer pool.

Normal Response Code: 201

Error Response Codes: Unauthorized (401), Bad Request (400)

This operation requires a request body.

This operation returns a response body.

Example 4.82. Create pool: HTTP request

```
POST /v2.0/lb/pools.json HTTP/1.1
User-Agent: python-neutronclient
Accept: application/json
```

```
{
   "pool":{
      "subnet_id":"8032909d-47a1-4715-90af-5153ffe39861",
      "lb_method":"ROUND_ROBIN",
      "protocol":"TCP",
      "name":"NewPool",
      "admin_state_up":true
   }
}
```

Example 4.83. Create pool: HTTP and JSON response

```
HTTP/1.1 201 Created
Content-Type: application/json; charset=UTF-8
```

```
{
   "pool":{
      "status":"PENDING_CREATE",
      "lb_method":"STATIC_IP",
      "protocol":"TCP",
```

```
      "description":"",
      "health_monitors":[

      ],
      "subnet_id":"8032909d-47a1-4715-90af-5153ffe39861",
      "tenant_id":"83657cfcdfe44cd5920adaf26c48ceea",
      "admin_state_up":true,
      "name":"NewPool",
      "members":[

      ],
      "id":"a224402b-794b-4c0c-9d08-d95640a6f5a1",
      "vip_id":null
   }
}
```

Update Pool

Verb	URI	Description
PUT	/lb/pools/*pool-id*	Updates a load balancer pool.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Bad Request (400), Not Found (404)

Example 4.84. Update pool: HTTP and JSON request

```
PUT /v2.0/lb/pools/61b1f87a-7a21-4ad3-9dda-7f81d249944f.json
User-Agent: python-neutronclient
Accept: application/json
```

```
{
   "pool":{
      "name":"SuperPool"
   }
}
```

Example 4.85. Update pool: HTTP and JSON response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
```

```
{
   "pool":{
      "status":"PENDING_UPDATE",
      "lb_method":"ROUND_ROBIN",
      "protocol":"TCP",
      "description":"",
      "health_monitors":[

      ],
      "subnet_id":"8032909d-47a1-4715-90af-5153ffe39861",
      "tenant_id":"83657cfcdfe44cd5920adaf26c48ceea",
      "admin_state_up":true,
      "name":"SuperPool",
      "members":[
```

```
      ],
      "id":"61b1f87a-7a21-4ad3-9dda-7f81d249944f",
      "vip_id":null
   }
}
```

Delete pool

Verb	URI	Description
DELETE	/lb/pools/*pool-id*	Removes a load balancer pool.

Normal Response Code: 204

Error Response Codes: Unauthorized (401), Not Found (404), Conflict (409)

This operation does not require a request body.

This operation does not return a response body.

Example 4.86. Delete pool: HTTP request

```
DELETE /v2.0/lb/pools/a224402b-794b-4c0c-9d08-d95640a6f5a1.json HTTP/1.1
Accept: application/json
```

Example 4.87. Delete pool: HTTP response

```
HTTP/1.1 204 No Content
Content-Length: 0
```

Member operations

This section discusses operations for managing pool members through the Load balancing as a service extension.

Table 4.8. Member Attributes

Attribute	Type	Required	CRUD [a]	Default value	Validation constraints	Notes
id	uuid-str	N/A	R	generated	N/A	Unique identifier for the member.
tenant_id	uuid-str	Yes	CR	Derived from authentication token.	N/A	Owner of the member. Only an admin user can specify a tenant identifier other than its own.
pool_id	uuid-str	Yes	CRU	None	UUID pattern.	The `pool` that the member belongs to.
address	IP	Yes	CR	None	IP address or null.	The IP address of the member.
protocol_port	Integer	Yes	CR	None	[0..65535]	The port on which the application is hosted.
weight	Integer	No	CRU	1	[0..256]	The weight of a member determines the portion of requests or connections it services compared to the other members of the

Attribute	Type	Required	CRUD [a]	Default value	Validation constraints	Notes
						pool. A value of 0 means the member does not participate in load-balancing but still accepts persistent connections.
admin_state_up	Bool	No	CRU	true	{true | false }	Administrative state of the member.
status	String	N/A	R	N/A	N/A	Indicates whether or not a member is currently operational.

[a] • **C**. Use the attribute in create operations.

- **R**. This attribute is returned in response to show and list operations.
- **U**. You can update the value of this attribute.
- **D**. You can delete the value of this attribute.

List members

Verb	URI	Description
GET	/lb/members	Lists members.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Forbidden (403)

This operation does not require a request body.

This operation returns a response body.

Example 4.88. List members: HTTP request

```
GET /v2.0/lb/members.json HTTP/1.1
User-Agent: python-neutronclient
Accept: application/json
```

Example 4.89. List members: HTTP and JSON response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
```


```
{
   "members":[
      {
         "status":"ACTIVE",
         "weight":1,
         "admin_state_up":true,
         "tenant_id":"83657cfcdfe44cd5920adaf26c48ceea",
         "pool_id":"72741b06-df4d-4715-b142-276b6bce75ab",
         "address":"10.0.0.4",
         "protocol_port":80,
         "id":"701b531b-111a-4f21-ad85-4795b7b12af6"
      },
      {
```

```
         "status":"ACTIVE",
         "weight":1,
         "admin_state_up":true,
         "tenant_id":"83657cfcdfe44cd5920adaf26c48ceea",
         "pool_id":"72741b06-df4d-4715-b142-276b6bce75ab",
         "address":"10.0.0.3",
         "protocol_port":80,
         "id":"beb53b4d-230b-4abd-8118-575b8fa006ef"
      }
   ]
}
```

Show member

Verb	URI	Description
GET	/lb/members/*member-id*	Returns details about a specific member.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Forbidden (403), Not Found (404)

This operation does not require a request body.

This operation returns a response body.

Example 4.90. Show member: HTTP request

```
GET /v2.0/lb/members/701b531b-111a-4f21-ad85-4795b7b12af6.json HTTP/1.1
User-Agent: python-neutronclient
Accept: application/json
```

Example 4.91. Show member: HTTP and JSON response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
```

```
{
   "member":{
      "status":"PENDING_CREATE",
      "protocol_port":8080,
      "weight":1,
      "admin_state_up":true,
      "tenant_id":"4fd44f30292945e481c7b8a0c8908869",
      "pool_id":"7803631d-f181-4500-b3a2-1b68ba2a75fd",
      "address":"10.0.0.5",
      "status_description":null,
      "id":"48a471ea-64f1-4eb6-9be7-dae6bbe40a0f"
   }
}
```

Create member

Verb	URI	Description
POST	/lb/members	Creates a load balancer member.

Normal Response Code: 201

Error Response Codes: Unauthorized (401), Bad Request (400)

This operation requires a request body.

This operation returns a response body.

Example 4.92. Create member: HTTP and JSON request

```
POST /v2.0/lb/members.json HTTP/1.1
User-Agent: python-neutronclient
Accept: application/json
```

```
{
   "member":{
      "protocol_port":"8080",
      "address":"10.0.0.5",
      "pool_id":"7803631d-f181-4500-b3a2-1b68ba2a75fd",
      "admin_state_up":true
   }
}
```

Example 4.93. Create member: HTTP and JSON response

```
HTTP/1.1 201 Created
Content-Type: application/json; charset=UTF-8
```

```
{
   "member":{
      "status":"PENDING_CREATE",
      "protocol_port":8080,
      "weight":1,
      "admin_state_up":true,
      "tenant_id":"4fd44f30292945e481c7b8a0c8908869",
      "pool_id":"7803631d-f181-4500-b3a2-1b68ba2a75fd",
      "address":"10.0.0.5",
      "status_description":null,
      "id":"48a471ea-64f1-4eb6-9be7-dae6bbe40a0f"
   }
}
```

Update member

Verb	URI	Description
PUT	/lb/members/*member-id*	Updates a load balancer member.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Bad Request (400), Not Found (404)

Example 4.94. Update member: HTTP and JSON request

```
PUT /v2.0/lb/members/b9a7012a-1097-45b2-a973-6572973619bc.json
User-Agent: python-neutronclient
```

```
Accept: application/json
```

```
{
   "member":{
      "admin_state_up":"False"
   }
}
```

Example 4.95. Update member: HTTP and JSON response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
```

```
{
   "member":{
      "status":"PENDING_UPDATE",
      "protocol_port":8080,
      "weight":1,
      "admin_state_up":false,
      "tenant_id":"4fd44f30292945e481c7b8a0c8908869",
      "pool_id":"7803631d-f181-4500-b3a2-1b68ba2a75fd",
      "address":"10.0.0.5",
      "status_description":null,
      "id":"48a471ea-64f1-4eb6-9be7-dae6bbe40a0f"
   }
}
```

Delete member

Verb	URI	Description
DELETE	/lb/members/*member-id*	Removes a load balancer member.

Normal Response Code: 204

Error Response Codes: Unauthorized (401), Not Found (404), Conflict (409)

This operation does not require a request body.

This operation does not return a response body.

Example 4.96. Delete member: HTTP request

```
DELETE /v2.0/lb/members/b9a7012a-1097-45b2-a973-6572973619bc.json HTTP/1.1
Accept: application/json
```

Example 4.97. Delete member: HTTP and JSON response

```
HTTP/1.1 204 No Content
Content-Length: 0
```

Health monitor operations

This section discusses operations for managing load balancer health monitors through the LBaaS extension.

Table 4.9. Health monitor attributes

Attribute	Type	Required	CRUD [a]	Default value	Validation constraints	Notes
id	uuid-str	N/A	R	generated	N/A	Unique ID for the health monitor.
tenant_id	uuid-str	Yes	CR	Derived from authentication token.	N/A	Owner of the health monitor. Only an admin user can specify a tenant identifier other than its own.
type	String	Yes	CR	None	{"PING" | "TCP" | "HTTP" | "HTTPS"}	The type of probe send by load balancer to verify member state
delay	Integer	Yes	CRU	None	non-negative	The time in seconds between sending probes to members.
timeout	uuid-str	Yes	CRU	None	non-negative	The maximum number of seconds for a monitor to wait for a connection to be established before it times out. The value must be less than the delay value.
max_retries	Integer	Yes	CRU	None	[1..10]	Number of allowed connection failures before changing the member's status to INACTIVE.
http_method	String	No [b]	CRU	GET	None	The HTTP method used for requests by the monitor.
url_path	String	No [b]	CRU	/	None	The HTTP path of the request sent by the monitor to test a member's health. This must be a string beginning with a / (forward slash).
expected_codes	String	No [b]	CRU	200	Single value, such as 200, list, such as 200, 202, or range, such as 200-204.	The list of HTTP status codes expected in response from the member to declare it healthy.
admin_state_up	Bool	No	CRU	true	{true | false }	Administrative state of the health monitor.
status	String	N/A	R	N/A	N/A	Indicates whether or not a health monitor is currently operational.

[a] • **C**. Use the attribute in create operations.

- **R**. This attribute is returned in response to show and list operations.
- **U**. You can update the value of this attribute.
- **D**. You can delete the value of this attribute.

[b] Required if `type` is `HTTP` or `HTTPS`.

List health monitors

Verb	URI	Description
GET	/lb/health_monitors	Lists health monitors.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Forbidden (403)

This operation does not require a request body.

This operation returns a response body.

Example 4.98. List health monitors: HTTP request

```
GET /v2.0/lb/health_monitors.json HTTP/1.1
User-Agent: python-neutronclient
Accept: application/json
```

Example 4.99. List health monitors: HTTP and JSON response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
```


```
{
   "health_monitors":[
      {
         "admin_state_up":true,
         "tenant_id":"83657cfcdfe44cd5920adaf26c48ceea",
         "delay":10,
         "max_retries":1,
         "timeout":1,
         "type":"PING",
         "id":"466c8345-28d8-4f84-a246-e04380b0461d"
      },
      {
         "admin_state_up":true,
         "tenant_id":"83657cfcdfe44cd5920adaf26c48ceea",
         "delay":5,
         "expected_codes":"200",
         "max_retries":2,
         "http_method":"GET",
         "timeout":2,
         "url_path":"/",
         "type":"HTTP",
         "id":"5d4b5228-33b0-4e60-b225-9b727c1a20e7"
      }
   ]
}
```


Show health monitor

Verb	URI	Description
GET	/lb/health_monitors/*health_monitor-id*	Show details for a specified health monitor.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Forbidden (403), Not Found (404)

This operation does not require a request body.

This operation returns a response body.

Example 4.100. Show health monitor: HTTP request

```
GET /v2.0/lb/health_monitors/701b531b-111a-4f21-ad85-4795b7b12af6.json HTTP/1.1
User-Agent: python-neutronclient
Accept: application/json
```

Example 4.101. Show health monitor: HTTP and JSON response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
```

```
{
    "health_monitor": {
        "admin_state_up": true,
        "tenant_id": "83657cfcdfe44cd5920adaf26c48ceea",
        "delay": 5,
        "expected_codes": "200",
        "max_retries": 2,
        "http_method": "GET",
        "timeout": 2,
        "url_path": "/",
        "type": "HTTP",
        "id": "5d4b5228-33b0-4e60-b225-9b727c1a20e7"
    }
}
```

Create health monitor

Verb	URI	Description
POST	/lb/health_monitors	Creates a load balancer health monitor.

Normal Response Code: 201

Error Response Codes: Unauthorized (401), Bad Request (400)

This operation requires a request body.

This operation returns a response body.

Example 4.102. Create health monitor: HTTP and JSON request

```
POST /v2.0/lb/health_monitors.json HTTP/1.1
User-Agent: python-neutronclient
Accept: application/json
```

```
{
   "health_monitor":{
      "delay":"1",
      "max_retries":"1",
      "type":"HTTP",
      "timeout":"1",
      "admin_state_up":true
   }
}
```

Example 4.103. Create health monitor: HTTP and JSON response

```
HTTP/1.1 201 Created
Content-Type: application/json; charset=UTF-8
```

```
{
   "health_monitor":{
      "admin_state_up":true,
      "tenant_id":"4fd44f30292945e481c7b8a0c8908869",
      "delay":1,
      "expected_codes":"200",
      "max_retries":1,
      "http_method":"GET",
      "timeout":1,
      "pools":[

      ],
      "url_path":"/",
      "type":"HTTP",
      "id":"b624decf-d5d3-4c66-9a3d-f047e7786181"
   }
}
```

Update health monitor

Verb	URI	Description
PUT	/lb/health_monitors/*health_monitor-id*	Updates a load balancer health monitor.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Bad Request (400), Not Found (404)

Example 4.104. Update health monitor: HTTP and JSON request

```
PUT /v2.0/lb/health_monitors/466c8345-28d8-4f84-a246-e04380b0461d.json HTTP/1.1
User-Agent: python-neutronclient
Accept: application/json
```

```
{
   "health_monitor":{
      "delay":"3"
   }
}
```

Example 4.105. Update health monitor: HTTP and JSON response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
```

```
{
   "health_monitor":{
      "admin_state_up":true,
      "tenant_id":"4fd44f30292945e481c7b8a0c8908869",
```

```
      "delay":3,
      "expected_codes":"200",
      "max_retries":1,
      "http_method":"GET",
      "timeout":1,
      "pools":[

      ],
      "url_path":"/",
      "type":"HTTP",
      "id":"b624decf-d5d3-4c66-9a3d-f047e7786181"
   }
}
```

Delete health monitor

Verb	URI	Description
DELETE	/lb/health_monitors/*healthmonitor-id*	Deletes a specified health monitor.

Normal Response Code: 204

Error Response Codes: Unauthorized (401), Not Found (404), Conflict (409)

This operation does not require a request body.

This operation does not return a response body.

Example 4.106. Delete health monitor: HTTP request

```
DELETE /v2.0/lb/health_monitors/ada338f0-10d0-4326-b02e-cc123114f240.json HTTP/1.1
Accept: application/json
```

Example 4.107. Delete health monitor: HTTP response

```
HTTP/1.1 204 No Content
Content-Length: 0
```

Associate health monitor with a pool

Verb	URI	Description
POST	/lb/pools/*pool-id*/health_monitors	Associate health monitor with the pool.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Bad Request (400), Not Found (404)

Example 4.108. Associate health monitor: HTTP and JSON request

```
POST /v2.0/lb/pools/72741b06-df4d-4715-b142-276b6bce75ab/health_monitors.json HTTP/1.1
User-Agent: python-neutronclient
Accept: application/json
```

```
{
   "health_monitor":{
      "id":"b624decf-d5d3-4c66-9a3d-f047e7786181"
   }
}
```

Example 4.109. Associate health monitor: HTTP and JSON response

```
HTTP/1.1 201 Created
Content-Type: application/json; charset=UTF-8
```

```
{
   "health_monitor":{

   }
}
```

Disassociate health monitor from a pool

Verb	URI	Description
DELETE	/lb/pools/*pool-id*/health_monitors/*healthmonitor-id*	Disassociates a health monitor from the pool.

Normal Response Code: 204

Error Response Codes: Unauthorized (401), Not Found (404), Conflict (409)

This operation does not require a request body.

This operation does not return a response body.

Example 4.110. Disassociate health monitor: HTTP request

```
DELETE /v2.0/lb/pools/72741b06-df4d-4715-b142-276b6bce75ab/health_monitors/ada338f0-10d0-4326-b0
Accept: application/json
```

Example 4.111. Disassociate health monitor: HTTP response

```
HTTP/1.1 204 No Content
Content-Length: 0
```

Firewall as a Service (FWaaS) Extension

The FWaaS extension provides OpenStack users with the ability to deploy firewalls to protect their networks. The current features provided by the FWaaS extension are:

- Apply firewall rules on traffic entering and leaving tenant networks.
- Support for applying tcp, udp, icmp, or protocol agnostic rules.
- Creation and sharing of firewall policies which hold an ordered collection of the firewall rules.
- Ability to audit firewall rules and policies.

This extension introduces new resources:

- **firewall**: represents a logical firewall resource that a tenant can instantiate and manage. A firewall is associated with one firewall_policy.
- **firewall_policy**: is an ordered collection of firewall_rules. A firewall_policy can be shared across tenants. Thus it can also be made part of an audit workflow wherein the firewall_policy can be audited by the relevant entity that is authorized (and can be different from the tenants which create or use the firewall_policy).
- **firewall_rule**: represents a collection of attributes like ports, ip addresses which define match criteria and action (allow, or deny) that needs to be taken on the matched data traffic.

Firewall Rule Operations

This section discusses operations for managing a Firewall Rule through this extension.

Table 4.10. Firewall Rule Attributes

Attribute	Type	Required	CRUD [a]	Default Value	Validation Constraints	Notes
id	uuid-str	N/A	R	generated	N/A	Unique identifier for the Firewall Rule object.
tenant_id	uuid-str	Yes	CR	Derived from Authentication token	N/A	Owner of the Firewall Rule. Only admin users can specify a tenant identifier other than their own.
name	String	No	CRU	None	N/A	Human readable name for the Firewall Rule (255 characters limit). Does not have to be unique.
description	String	No	CRU	None	N/A	Human readable description for the Firewall Rule (1024 characters limit).
firewall_policy_id	uuid-str	No	R	None	N/A	This is a readonly attribute which gets populated with the uuid of the Firewall Policy when this Firewall Rule is associated with a Firewall Policy. A Firewall Rule can be associated with one Firewall Policy at a time. The association can however be updated to a different Firewall Policy. This attribute can be "null" if the rule is not associated with any firewall policy.
shared	Bool	No	CRU	false	{true | false}	When set to True makes this Firewall Rule visible to tenants other than its owner, and can be used in Firewall Policies not owned by its tenant.
protocol	String	No	CRU	None	{icmp | tcp | udp | null}	IP Protocol
ip_version	Integer	No	CRU	4	{4 | 6}	IP Protocol Version
source_ip_address	String (IP address or CIDR)	No	CRU	None	valid IP address (v4 or v6), or CIDR	Source IP address or CIDR
destination_ip_address	String (IP address or CIDR)	No	CRU	None	Valid IP address (v4 or v6), or CIDR	Destination IP address or CIDR
source_port	Integer	No	CRU	None	Valid port number (integer or string), or port range in the format of a ':' separated	Source port number or a range

Attribute	Type	Required	CRUD [a]	Default Value	Validation Constraints	Notes
					range). In the case of port range, both ends of the range are included.	
destination_port	Integer	No	CRU	None	Valid port number (integer or string), or port range in the format of a ':' separated range. In the case of port range, both ends of the range are included.	Destination port number or a range
position	Integer	No	R	None	N/A	This is a readonly attribute that gets assigned to this rule when the rule is associated with a Firewall Policy. It indicates the position of this rule in that Firewall Policy. This position number starts at 1. The position can be "null" if the firewall rule is not associated with any policy.
action	String	No	CRU	deny	{allow | deny}	Action to be performed on the traffic matching the rule (allow, deny)
enabled	Bool	No	CRU	true	{true | false}	When set to False will disable this rule in the Firewall Policy. Facilitates selectively turning off rules without having to disassociate the rule from the Firewall Policy

[a]

- **C**. Use the attribute in create operations.
- **R**. This attribute is returned in response to show and list operations.
- **U**. You can update the value of this attribute.
- **D**. You can delete the value of this attribute.

List Firewall Rules

Verb	URI	Description
GET	/fw/firewall_rules	List Firewall Rules.

Normal Response Code: 200

Error Response Codes: Unauthorized (401).

This operation does not require a request body.

This operation returns a response body.

Example 4.112. List Firewall Rules: Request

```
GET /v2.0/fw/firewall_rules.json
User-Agent: python-neutronclient
Accept: application/json
```

Example 4.113. List Firewall Rules: Response

```
{
    "firewall_rules": [
        {
            "action": "allow",
            "description": "",
            "destination_ip_address": null,
            "destination_port": "80",
            "enabled": true,
            "firewall_policy_id": "c69933c1-b472-44f9-8226-30dc4ffd454c",
            "id": "8722e0e0-9cc9-4490-9660-8c9a5732fbb0",
            "ip_version": 4,
            "name": "ALLOW_HTTP",
            "position": 1,
            "protocol": "tcp",
            "shared": false,
            "source_ip_address": null,
            "source_port": null,
            "tenant_id": "45977fa2dbd7482098dd68d0d8970117"
        }
    ]
}
```

Show Firewall Rule

Verb	URI	Description
GET	/fw/firewall_rules/*firewall_rule-id*	Returns details about a specific Firewall Rule.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Forbidden (403), Not Found (404)

This operation does not require a request body.

This operation returns a response body.

Example 4.114. Show Firewall Rule: Request

```
GET /v2.0/fw/firewall_rules/9faaf49f-dd89-4e39-a8c6-101839aa49bc.json
User-Agent: python-neutronclient
Accept: application/json
```

Example 4.115. Show Firewall Rule: Response

```
{
    "firewall_rule": {
        "action": "allow",
        "description": "",
        "destination_ip_address": null,
        "destination_port": "80",
        "enabled": true,
        "firewall_policy_id": null,
        "id": "8722e0e0-9cc9-4490-9660-8c9a5732fbb0",
        "ip_version": 4,
        "name": "ALLOW_HTTP",
        "position": null,
        "protocol": "tcp",
        "shared": false,
        "source_ip_address": null,
        "source_port": null,
        "tenant_id": "45977fa2dbd7482098dd68d0d8970117"
    }
}
```

Create Firewall Rule

Verb	URI	Description
POST	/fw/firewall_rules	Creates a new Firewall Rule.

Normal Response Code: 201

Error Response Codes: Unauthorized (401), Bad Request (400)

This operation requires a request body.

This operation returns a response body.

Example 4.116. Create Firewall Rule: Request

```
POST /v2.0/fw/firewall_rules.json
User-Agent: python-neutronclient
Accept: application/json
```

```
{
    "firewall_rule": {
        "action": "allow",
        "destination_port": "80",
        "enabled": true,
        "name": "ALLOW_HTTP",
        "protocol": "tcp"
    }
}
```

Example 4.117. Create Firewall Rule: Response

```
HTTP/1.1 201 Created
Content-Type: application/json; charset=UTF-8
```

```
{
    "firewall_rule": {
        "action": "allow",
        "description": "",
        "destination_ip_address": null,
        "destination_port": "80",
        "enabled": true,
        "firewall_policy_id": null,
        "id": "8722e0e0-9cc9-4490-9660-8c9a5732fbb0",
        "ip_version": 4,
        "name": "ALLOW_HTTP",
        "position": null,
        "protocol": "tcp",
        "shared": false,
        "source_ip_address": null,
        "source_port": null,
        "tenant_id": "45977fa2dbd7482098dd68d0d8970117"
    }
}
```

Update Firewall Rule

Verb	URI	Description
PUT	/fw/firewall_rules/*firewall_rule-id*	Updates a Firewall Rule.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Bad Request (400), Not Found (404)

Example 4.118. Update Firewall Rule: Request

```
PUT /v2.0/fw/firewall_rules/41bfef97-af4e-4f6b-a5d3-4678859d2485.json
User-Agent: python-neutronclient
Accept: application/json
```

```
{
    "firewall_rule": {
        "shared": "true"
    }
}
```

Example 4.119. Update Firewall Rule: Response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
```

```
{
    "firewall_rule": {
        "action": "allow",
        "description": "",
        "destination_ip_address": null,
        "destination_port": "80",
        "enabled": true,
        "firewall_policy_id": "c69933c1-b472-44f9-8226-30dc4ffd454c",
        "id": "8722e0e0-9cc9-4490-9660-8c9a5732fbb0",
        "ip_version": 4,
        "name": "ALLOW_HTTP",
        "position": 1,
        "protocol": "tcp",
        "shared": true,
        "source_ip_address": null,
        "source_port": null,
        "tenant_id": "45977fa2dbd7482098dd68d0d8970117"
    }
}
```

Delete Firewall Rule

Verb	URI	Description
DELETE	/fw/ firewall_rules/ *firewall_rule-id*	Removes a Firewall Rule.

Normal Response Code: 204

Error Response Codes: Unauthorized (401), Not Found (404), Conflict (409). The Conflict error response is returned when an operation is performed while the firewall is in a PENDING state.

This operation does not require a request body.

This operation does not return a response body.

Example 4.120. Delete Firewall Rule: Request

```
DELETE /v2.0/fw/firewall_rules/1be5e5f7-c45e-49ba-85da-156575b60d50.json
User-Agent: python-neutronclient
Accept: application/json
```

Example 4.121. Delete Firewall Rule: Response

```
HTTP/1.1 204 No Content
Content-Length: 0
```

Firewall Policy Operations

This section discusses operations for managing a Firewall Policy through this extension.

Table 4.11. Firewall Policy Attributes

Attribute	Type	Required	CRUD [a]	Default Value	Validation Constraints	Notes
id	uuid-str	N/A	R	generated	N/A	Unique identifier for the Firewall Policy object.
tenant_id	uuid-str	Yes	CR	Derived from Authentication token	N/A	Owner of the Firewall Policy. Only admin users can specify a tenant identifier other than their own.
name	String	No	CRU	None	N/A	Human readable name for the Firewall Policy (255 characters limit). Does not have to be unique.
description	String	No	CRU	None	N/A	Human readable description for the Firewall Policy (1024 characters limit)
shared	Bool	No	CRU	false	{true | false}	When set to True makes this Firewall Policy visible to tenants other than its owner.
firewall_rules	List	No	CRU	Empty list	JSON list of Firewall Rule uuids	This is an ordered list of Firewall Rule uuids. The Firewall applies the rules in the order in which they appear in this list.
audited	Bool	No	CRU	false	{true | false}	When set to True by the policy owner indicates that the Firewall Policy has been audited. This attribute is meant to aid in the firewall policy audit workflows. Each time the Firewall Policy or the associated Firewall Rules are changed, this attribute will be set to False and will have to be explicitly set to True through an update operation.

[a]

- **C**. Use the attribute in create operations.
- **R**. This attribute is returned in response to show and list operations.
- **U**. You can update the value of this attribute.
- **D**. You can delete the value of this attribute.

List Firewall Policies

Verb	URI	Description
GET	/fw/firewall_policies	List Firewall Policies.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Forbidden (403)

This operation does not require a request body.

This operation returns a response body.

Example 4.122. List Firewall Policies: Request

```
GET /v2.0/fw/firewall_policies.json
User-Agent: python-neutronclient
Accept: application/json
```

Example 4.123. List Firewall Policies: Response

```
{
    "firewall_policies": [
        {
            "audited": false,
            "description": "",
            "firewall_rules": [
                "8722e0e0-9cc9-4490-9660-8c9a5732fbb0"
            ],
            "id": "c69933c1-b472-44f9-8226-30dc4ffd454c",
            "name": "test-policy",
            "shared": false,
            "tenant_id": "45977fa2dbd7482098dd68d0d8970117"
        }
    ]
}
```

Show Firewall Policy

Verb	URI	Description
GET	/fw/firewall_policies/*firewall_policy-id*	Returns details about a specific Firewall Policy.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Not Found (404)

This operation does not require a request body.

This operation returns a response body.

Example 4.124. Show Firewall Policy: Request

```
GET /v2.0/fw/firewall_policies/9faaf49f-dd89-4e39-a8c6-101839aa49bc.json
User-Agent: python-neutronclient
Accept: application/json
```

Example 4.125. Show Firewall Policy: Response

```
{
    "firewall_policy": {
        "audited": false,
        "description": "",
        "firewall_rules": [
            "8722e0e0-9cc9-4490-9660-8c9a5732fbb0"
        ],
        "id": "c69933c1-b472-44f9-8226-30dc4ffd454c",
        "name": "test-policy",
        "shared": false,
        "tenant_id": "45977fa2dbd7482098dd68d0d8970117"
    }
}
```

Create Firewall Policy

Verb	URI	Description
POST	/fw/firewall_policies	Creates a new Firewall Policy.

Normal Response Code: 201

Error Response Codes: Unauthorized (401).

This operation requires a request body.

This operation returns a response body.

Example 4.126. Create Firewall Policy: Request

```
POST /v2.0/fw/firewall_policies.json
User-Agent: python-neutronclient
Accept: application/json
```

```
{
    "firewall_policy": {
        "firewall_rules": [
            "8722e0e0-9cc9-4490-9660-8c9a5732fbb0"
        ],
        "name": "test-policy"
    }
}
```

Example 4.127. Create Firewall Policy: Response

```
HTTP/1.1 201 Created
Content-Type: application/json; charset=UTF-8
```

```
{
    "firewall_policy": {
        "audited": false,
        "description": "",
        "firewall_rules": [
            "8722e0e0-9cc9-4490-9660-8c9a5732fbb0"
        ],
        "id": "c69933c1-b472-44f9-8226-30dc4ffd454c",
        "name": "test-policy",
        "shared": false,
        "tenant_id": "45977fa2dbd7482098dd68d0d8970117"
    }
}
```

Update Firewall Policy

Verb	URI	Description
PUT	/fw/firewall_policies/*firewall_policy-id*	Updates a Firewall Policy.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Not Found (404)

Example 4.128. Update Firewall Policy: Request

```
PUT /v2.0/fw/firewall_policies/41bfef97-af4e-4f6b-a5d3-4678859d2485.json
User-Agent: python-neutronclient
Accept: application/json
```

```
{
    "firewall_policy": {
        "firewall_rules": [
            "a08ef905-0ff6-4784-8374-175fffe7dade",
            "8722e0e0-9cc9-4490-9660-8c9a5732fbb0"
        ]
    }
}
```

Example 4.129. Update Firewall Policy: Response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
```

```
{
    "firewall_policy": {
        "audited": false,
        "description": "",
        "firewall_rules": [
            "a08ef905-0ff6-4784-8374-175fffe7dade",
            "8722e0e0-9cc9-4490-9660-8c9a5732fbb0"
        ],
        "id": "c69933c1-b472-44f9-8226-30dc4ffd454c",
        "name": "test-policy",
        "shared": false,
        "tenant_id": "45977fa2dbd7482098dd68d0d8970117"
    }
}
```

Delete Firewall Policy

Verb	URI	Description
DELETE	/fw/firewall_policies/*firewall_policy-id*	Removes a Firewall Policy.

Normal Response Code: 204

Error Response Codes: Unauthorized (401), Not Found (404), Conflict (409). Conflict error code is returned the firewall policy is in use.

This operation does not require a request body.

This operation does not return a response body.

Example 4.130. Delete Firewall Policy: Request

```
DELETE /v2.0/fw/firewall_policies/1be5e5f7-c45e-49ba-85da-156575b60d50.json
User-Agent: python-neutronclient
Accept: application/json
```

Example 4.131. Delete Firewall Policy: Response

```
HTTP/1.1 204 No Content
Content-Length: 0
```

Positionally inserting a Firewall Rule in a Firewall Policy

Verb	URI	Description
PUT	/fw/firewall_policies/*firewall_policy-id*/insert_rule	Inserts a Firewall Rule in a Firewall Policy relative to the position of other rules.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Bad Request (400), Not Found (404). Bad Request error is returned in the case the rule information is missing.

Example 4.132. Insert Firewall Rule in Firewall Policy: Request

```
PUT /v2.0/fw/firewall_policies/41bfef97-af4e-4f6b-a5d3-4678859d2485/
insert_rule.json
User-Agent: python-neutronclient
Accept: application/json
```

```
{
    "firewall_rule_id": "7bc34b8c-8d3b-4ada-a9c8-1f4c11c65692",
    "insert_after": "a08ef905-0ff6-4784-8374-175fffe7dade",
    "insert_before": ""
}
```

Example 4.133. Insert Firewall Rule in Firewall Policy: Response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
```

```
{
    "audited": false,
    "description": "",
    "firewall_list": [],
    "firewall_rules": [
        "a08ef905-0ff6-4784-8374-175fffe7dade",
        "7bc34b8c-8d3b-4ada-a9c8-1f4c11c65692",
        "8722e0e0-9cc9-4490-9660-8c9a5732fbb0"
    ],
    "id": "c69933c1-b472-44f9-8226-30dc4ffd454c",
    "name": "test-policy",
    "shared": false,
    "tenant_id": "45977fa2dbd7482098dd68d0d8970117"
}
```

insert_before and insert_after parameters refer to firewall rule uuids already associated with the firewall policy. firewall_rule_id refers to uuid of the rule being inserted. insert_before takes precedence over insert_after and if neither is specified, firewall_rule_is is inserted at the first position.

Removing a Firewall Rule from a Firewall Policy

Verb	URI	Description
PUT	/fw/firewall_policies/*firewall_policy-id*/remove_rule	Removes a Firewall Rule from a Firewall Policy.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Bad Request (400), Not Found (404). Bad Request error is returned if the rule information is missing or when a firewall rule is tried to be removed from a firewall policy to which it is not associated.

Example 4.134. Remove Firewall Rule from Firewall Policy: Request

```
PUT /v2.0/fw/firewall_policies/41bfef97-af4e-4f6b-a5d3-4678859d2485/
remove_rule.json
User-Agent: python-neutronclient
Accept: application/json
```

```
{
    "firewall_rule_id": "7bc34b8c-8d3b-4ada-a9c8-1f4c11c65692"
}
```

Example 4.135. Remove Firewall Rule from Firewall Policy: Response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
```

```
{
    "audited": false,
    "description": "",
    "firewall_list": [],
    "firewall_rules": [
        "a08ef905-0ff6-4784-8374-175fffe7dade",
        "8722e0e0-9cc9-4490-9660-8c9a5732fbb0"
    ],
    "id": "c69933c1-b472-44f9-8226-30dc4ffd454c",
    "name": "test-policy",
    "shared": false,
    "tenant_id": "45977fa2dbd7482098dd68d0d8970117"
}
```

Firewall Operations

This section discusses operations for managing a Firewall through this extension.

Table 4.12. Firewall Attributes

Attribute	Type	Required	CRUD [a]	Default Value	Validation Constraints	Notes
id	uuid-str	N/A	R	generated	N/A	Unique identifier for the Firewall object.
tenant_id	uuid-str	Yes	CR	Derived from Authentication token	N/A	Owner of the Firewall. Only admin users can specify a tenant identifier other than their own.
name	String	No	CRU	None	N/A	Human readable name for the Firewall (255 characters limit). Does not have to be unique.
description	String	No	CRU	None	N/A	Human readable description for the Firewall (1024 characters limit)
admin_state_up	Bool	N/A	CRU	true	{true | false }	Administrative state of the Firewall. If false (down), firewall does not forward packets and will drop all traffic to/from VMs behind the firewall.
status	String	N/A	R	N/A	N/A	Indicates whether Firewall resource is currently operational. Possible values include: ACTIVE, DOWN, BUILD, ERROR, PENDING_CREATE, PENDING_UPDATE, or PENDING_DELETE.
shared	Bool	No	CRU	false	{true | false}	When set to True makes this Firewall Rule visible to tenants other than its owner, and can be used in Firewall Policies not owned by its tenant.
firewall_policy_id	uuid-str	No	CRU	None	valid Firewall Policy uuid	The Firewall Policy uuid that this Firewall is associated with. This Firewall will implement the rules contained in the Firewall Policy represented by this uuid.

[a]

- **C**. Use the attribute in create operations.
- **R**. This attribute is returned in response to show and list operations.
- **U**. You can update the value of this attribute.
- **D**. You can delete the value of this attribute.

List Firewalls

Verb	URI	Description
GET	/fw/firewalls	List Firewalls.

Normal Response Code: 200

Error Response Codes: Unauthorized (401)

This operation does not require a request body.

This operation returns a response body.

Example 4.136. List Firewalls: Request

```
GET /v2.0/fw/firewalls.json
User-Agent: python-neutronclient
Accept: application/json
```

Example 4.137. List Firewalls: Response

```
{
    "firewalls": [
        {
            "admin_state_up": true,
            "description": "",
            "firewall_policy_id": "c69933c1-b472-44f9-8226-30dc4ffd454c",
            "id": "3b0ef8f4-82c7-44d4-a4fb-6177f9a21977",
            "name": "",
            "status": "ACTIVE",
            "tenant_id": "45977fa2dbd7482098dd68d0d8970117"
        }
    ]
}
```

Show Firewall

Verb	URI	Description
GET	/fw/firewalls/*firewall-id*	Returns details about a specific Firewall.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Forbidden (403), Not Found (404)

This operation does not require a request body.

This operation returns a response body.

Example 4.138. Show Firewall: Request

```
GET /v2.0/fw/firewalls/9faaf49f-dd89-4e39-a8c6-101839aa49bc.json
User-Agent: python-neutronclient
Accept: application/json
```

Example 4.139. Show Firewall: Response

```
{
    "firewall": {
        "admin_state_up": true,
        "description": "",
        "firewall_policy_id": "c69933c1-b472-44f9-8226-30dc4ffd454c",
        "id": "3b0ef8f4-82c7-44d4-a4fb-6177f9a21977",
        "name": "",
        "status": "ACTIVE",
        "tenant_id": "45977fa2dbd7482098dd68d0d8970117"
    }
}
```

Create Firewall

Verb	URI	Description
POST	/fw/firewalls	Creates a new Firewall.

Normal Response Code: 201

Error Response Codes: Unauthorized (401), Bad Request (400)

This operation requires a request body.

This operation returns a response body.

Example 4.140. Create Firewall: Request

```
POST /v2.0/fw/firewalls.json
User-Agent: python-neutronclient
Accept: application/json
```

```
{
    "firewall": {
        "admin_state_up": true,
        "firewall_policy_id": "c69933c1-b472-44f9-8226-30dc4ffd454c"
    }
}
```

Example 4.141. Create Firewall: Response

```
HTTP/1.1 201 Created
Content-Type: application/json; charset=UTF-8
```

```
{
    "firewall": {
        "admin_state_up": true,
        "description": "",
        "firewall_policy_id": "c69933c1-b472-44f9-8226-30dc4ffd454c",
        "id": "3b0ef8f4-82c7-44d4-a4fb-6177f9a21977",
        "name": "",
        "status": "PENDING_CREATE",
        "tenant_id": "45977fa2dbd7482098dd68d0d8970117"
    }
}
```

Update Firewall

Verb	URI	Description
PUT	/fw/firewalls/*firewall-id*	Updates a Firewall, provided status is not indicating a PENDING_* state.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Bad Request (400), Not Found (404)

Example 4.142. Update Firewall: Request

```
PUT /v2.0/fw/firewalls/41bfef97-af4e-4f6b-a5d3-4678859d2485.json
User-Agent: python-neutronclient
Accept: application/json
```

```
{
    "firewall": {
        "admin_state_up": "false"
    }
}
```

Example 4.143. Update Firewall: Response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
```

```
{
    "firewall": {
        "admin_state_up": false,
        "description": "",
        "firewall_policy_id": "c69933c1-b472-44f9-8226-30dc4ffd454c",
        "id": "3b0ef8f4-82c7-44d4-a4fb-6177f9a21977",
        "name": "",
        "status": "PENDING_UPDATE",
        "tenant_id": "45977fa2dbd7482098dd68d0d8970117"
    }
}
```

Delete Firewall

Verb	URI	Description
DELETE	/fw/firewalls/*firewall-id*	Removes a Firewall.

Normal Response Code: 204

Error Response Codes: Unauthorized (401), Not Found (404)

This operation does not require a request body.

This operation does not return a response body.

Example 4.144. Delete Firewall: Request

```
DELETE /v2.0/fw/firewalls/1be5e5f7-c45e-49ba-85da-156575b60d50.json
User-Agent: python-neutronclient
Accept: application/json
```

Example 4.145. Delete Firewall: Response

```
HTTP/1.1 204 No Content
Content-Length: 0
```

Agent Schedulers

The agent scheduler extensions schedule resources among agents on top of the agent management extension.

The agent scheduler feature consist of several agent scheduler extensions. In Havana, the following extensions are available.

- DHCP Agent Scheduler (`dhcp_agent_scheduler`)
- L3 Agent Scheduler (`l3_agent_scheduler`)
- Load Balancer Agent Scheduler (`lbaas_agent_scheduler`)

In Grizzly, the DHCP Agent Scheduler and the L3 Agent Scheduler features are provided by a single extension named the Agent Scheduler (`agent_scheduler`). In Havana, this extension is split into the DHCP Agent Scheduler and the L3 Agent Scheduler extensions. The Load Balancer Agent Scheduler extension is added in Havana.

DHCP Agent Scheduler (dhcp_agent_scheduler)

The DHCP Agent Scheduler extension allows administrators to assign DHCP servers for Neutron networks to given Neutron DHCP agents, and retrieve mappings between Neutron networks and DHCP agents. This feature is implemented on top of Agent Management extension.

Verb	URI	Description
GET	/agents/*agent_id*/dhcp-networks	List networks the given DHCP agent is hosting.
GET	/networks/*network_id*/dhcp-agents	List DHCP agents hosting the given network.
POST	/agents/*agent_id*/dhcp-networks	Schedule the network to the given DHCP agent.
DELETE	/agents/*agent_id*/dhcp-networks/*network_id*	Remove the network from the given DHCP agent.

List Networks hosted by one DHCP agent

Verb	URI	Description
GET	/agents/*agent_id*/dhcp-networks	List networks the given DHCP agent is hosting.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Forbidden (403)

This operation does not require a request body.

This operation returns a response body.

Example 4.146. List Networks on DHCP agent: JSON Request

```
GET /v2.0/agents/d5724d7e-389d-4ba0-8d00-fc673a147bfa/dhcp-networks HTTP/1.1
Host: localhost:9696
User-Agent: python-neutronclient
Content-Type: application/json
Accept: application/json
X-Auth-Token: 797f94caf0a8481c893a232cc0c1dfca
```

Example 4.147. List Networks on DHCP agent: JSON Response

```
{
   "networks":[
      {
         "status":"ACTIVE",
         "subnets":[
            "15a09f6c-87a5-4d14-b2cf-03d97cd4b456"
         ],
         "name":"net1",
         "provider:physical_network":"physnet1",
         "admin_state_up":true,
         "tenant_id":"3671f46ec35e4bbca6ef92ab7975e463",
         "provider:network_type":"vlan",
         "router:external":false,
         "shared":false,
         "id":"2d627131-c841-4e3a-ace6-f2dd75773b6d",
         "provider:segmentation_id":1001
      },
      {
         "status":"ACTIVE",
         "subnets":[

         ],
         "name":"net2",
         "provider:physical_network":null,
         "admin_state_up":true,
         "tenant_id":"3671f46ec35e4bbca6ef92ab7975e463",
         "provider:network_type":"local",
         "router:external":false,
         "shared":false,
         "id":"524e26ea-fad4-4bb0-b504-1ad0dc770e7a",
         "provider:segmentation_id":null
      },
      {
         "status":"ACTIVE",
         "subnets":[
            "43671fba-c76b-4c33-bd7e-8bef54145f2f"
         ],
         "name":"mynet1",
         "provider:physical_network":"physnet1",
         "admin_state_up":true,
         "tenant_id":"3671f46ec35e4bbca6ef92ab7975e463",
         "provider:network_type":"vlan",
         "router:external":false,
         "shared":false,
         "id":"cfa65a54-06a8-4f9f-86b0-73c700c02c41",
         "provider:segmentation_id":1000
```

```
      }
   ]
}
```

List DHCP agents hosting network

Verb	URI	Description
GET	/ networks/ *network_id*/dhcp-agents	List DHCP agents hosting the given network.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Forbidden (403)

This operation does not require a request body.

This operation returns a response body.

Example 4.148. List DHCP agents hosting network: JSON Request

```
GET /v2.0/networks/2d627131-c841-4e3a-ace6-f2dd75773b6d/dhcp-agents HTTP/1.1
Host: localhost:9696
User-Agent: python-neutronclient
Content-Type: application/json
Accept: application/json
X-Auth-Token: cc0f378bdf1545fb8dea2120c89eb532
```

Example 4.149. List DHCP agents hosting network: JSON Response

```
{
   "agents":[
      {
         "binary":"neutron-dhcp-agent",
         "description":null,
         "admin_state_up":true,
         "heartbeat_timestamp":"2013-03-27T00:24:01.000000",
         "alive":false,
         "topic":"dhcp_agent",
         "host":"HostC",
         "agent_type":"DHCP agent",
         "created_at":"2013-03-26T23:54:20.000000",
         "started_at":"2013-03-26T23:54:20.000000",
         "id":"d5724d7e-389d-4ba0-8d00-fc673a147bfa",
         "configurations":{
            "subnets":2,
            "use_namespaces":true,
            "dhcp_driver":"neutron.agent.linux.dhcp.Dnsmasq",
            "networks":2,
            "dhcp_lease_time":120,
            "ports":5
         }
      }
   ]
}
```

Schedule network to DHCP agent

Verb	URI	Description
POST	/agents/*agent_id*/ dhcp-networks	Schedule the network to the given DHCP agent.

Normal Response Code: 201

Error Response Codes: Unauthorized (401), Forbidden (403), Conflict (409) if the network is already hosted by the given DHCP agent, NotFound(404) when the specified agent is not a valid DHCP agent.

This operation requires a request body.

This operation returns a `null` body.

Example 4.150. Schedule Network: JSON Request

```
POST /v2.0/agents/d5724d7e-389d-4ba0-8d00-fc673a147bfa/dhcp-networks.json
 HTTP/1.1
Host: localhost:9696
User-Agent: python-neutronclient
Content-Type: application/json
Accept: application/json
X-Auth-Token: d88f7af21ee34f6c87e23e46cf3f986d
Content-Length: 54

{"network_id": "1ae075ca-708b-4e66-b4a7-b7698632f05f"}
```

Example 4.151. Schedule Network: JSON Response

```
HTTP/1.1 201 Created
Content-Type: application/json; charset=UTF-8
Content-Length: 4
Date: Wed, 27 Mar 2013 01:22:46 GMT

null
```

Remove Network From DHCP agent

Verb	URI	Description
DELETE	/agents/*agent_id*/ dhcp-networks/ *network_id*	Remove the network from the given DHCP agent.

Normal Response Code: 204

Error Response Codes: Unauthorized (401), Forbidden (403), NotFound (404), Conflict (409) if the network is not hosted by the given DHCP agent.

This operation does not require a request body.

This operation does not return a response body.

Example 4.152. Remove Network From DHCP agent: JSON Request

```
DELETE /v2.0/agents/d5724d7e-389d-4ba0-8d00-fc673a147bfa/dhcp-networks/
1ae075ca-708b-4e66-b4a7-b7698632f05f.json HTTP/1.1
Host: localhost:9696
User-Agent: python-neutronclient
Content-Type: application/json
Accept: application/json
X-Auth-Token: 7ae91cde8f504031be5a2cd5b99d4fe9
```

L3 Agent Scheduler (l3_agent_scheduler)

The L3 Agent Scheduler extension allows administrators to assign Neutron routers to Neutron L3 agents, and retrieve mappings between Neutron routers and L3 agents. This feature is implemented on top of Agent Management extension.

Verb	URI	Description
GET	/agents/*agent_id*/l3-routers	List routers the given L3 agent is hosting.
GET	/routers/*router_id*/l3-agents	List L3 agents hosting the given router.
POST	/agents/*agent_id*/l3-routers	Schedule the router to the given L3 agent.
DELETE	/agents/*agent_id*/l3-routers/*router_id*	Remove the router from the given L3 agent.

List Routers hosted by one L3 agent

Verb	URI	Description
GET	/agents/*agent_id*/l3-routers	List routers the given L3 agent is hosting.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Forbidden (403)

This operation does not require a request body.

This operation returns a response body.

Example 4.153. List Routers on L3 agent: JSON Request

```
GET /v2.0/agents/fa24e88e-3d2f-4fc2-b038-5fb5be294c03/l3-routers.json HTTP/1.1
Host: localhost:9696
User-Agent: python-neutronclient
Content-Type: application/json
Accept: application/json
X-Auth-Token: 6eeea6e73b68415f85d8368902a32c11
```

Example 4.154. List Routers on L3 agent: JSON Response

```
{
   "routers":[
      {
```

```
         "status":"ACTIVE",
         "external_gateway_info":null,
         "name":"router1",
         "admin_state_up":true,
         "tenant_id":"3671f46ec35e4bbca6ef92ab7975e463",
         "routes":[

         ],
         "id":"8eef2388-f27d-4a17-986e-9319a77ccd9d"
      }
   ]
}
```

List L3 agents hosting router

Verb	URI	Description
GET	/routers/*router_id*/l3-agents	List L3 agents hosting the given router.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Forbidden (403)

This operation does not require a request body.

This operation returns a response body.

Example 4.155. List L3 agents hosting router: JSON Request

```
GET /v2.0/routers/8eef2388-f27d-4a17-986e-9319a77ccd9d/l3-agents.json HTTP/1.1
Host: localhost:9696
User-Agent: python-neutronclient
Content-Type: application/json
Accept: application/json
X-Auth-Token: bce63afb1e794c70972a19a7c2d6dcab
```

Example 4.156. List L3 agents hosting router: JSON Response

```
{
   "agents":[
      {
         "binary":"neutron-l3-agent",
         "description":null,
         "admin_state_up":true,
         "heartbeat_timestamp":"2013-03-27T00:24:03.000000",
         "alive":false,
         "topic":"l3_agent",
         "host":"HostC",
         "agent_type":"L3 agent",
         "created_at":"2013-03-26T23:54:26.000000",
         "started_at":"2013-03-26T23:54:26.000000",
         "id":"fa24e88e-3d2f-4fc2-b038-5fb5be294c03",
         "configurations":{
            "router_id":"",
            "gateway_external_network_id":"",
            "handle_internal_only_routers":true,
            "use_namespaces":true,
            "routers":0,
```

```
            "interfaces":0,
            "floating_ips":0,
            "interface_driver":"neutron.agent.linux.interface.
OVSInterfaceDriver",
            "ex_gw_ports":0
         }
      }
   ]
}
```

Schedule router to L3 agent

Verb	URI	Description
POST	/agents/*agent_id*/ l3-routers	Schedule one router to the given L3 agent.

Normal Response Code: 201

Error Response Codes: Unauthorized (401), Forbidden (403), Conflict (409) if the router is already hosted, NotFound (404) if the specified agent is not a valid L3 agent.

This operation requires a request body.

This operation returns a `null` body.

Example 4.157. Schedule Router: JSON Request

```
GET /v2.0/agents/fa24e88e-3d2f-4fc2-b038-5fb5be294c03/l3-routers.json HTTP/1.1
Host: localhost:9696
User-Agent: python-neutronclient
Content-Type: application/json
Accept: application/json
X-Auth-Token: d88f7af21ee34f6c87e23e46cf3f986d
Content-Length: 54

{"router_id": "8eef2388-f27d-4a17-986e-9319a77ccd9d"}
```

Example 4.158. Schedule Router: JSON Response

```
HTTP/1.1 201 Created
Content-Type: application/json; charset=UTF-8
Content-Length: 4
Date: Wed, 27 Mar 2013 01:22:46 GMT

null
```

Remove Router From L3 agent

Verb	URI	Description
DELETE	/agents/*agent_id*/ l3-routers/*network_id*	Remove the router from the given L3 agent.

Normal Response Code: 204

Error Response Codes: Unauthorized (401), Forbidden (403), Conflict (409) if the router is not hosted by the given L3 agent.

This operation does not require a request body.

This operation does not return a response body.

Example 4.159. Remove Router From L3 agent: JSON Request

```
DELETE /v2.0/agents/b7d7ba43-1a05-4b09-ba07-67242d4a98f4/l3-routers/8eef2388-
f27d-4a17-986e-9319a77ccd9d.json HTTP/1.1
Host: localhost:9696
User-Agent: python-neutronclient
Content-Type: application/json
Accept: application/json
X-Auth-Token: 2147ef6fe4444f0299b1c0b6b529ff47
```

Load Balancer Agent Scheduler (lbaas_agent_scheduler)

The LBaaS Agent Scheduler extension allows administrators to retrieve mappings between load balancer pools to LBaaS agents. In Havana, this extension does not provide an ability to assign load balancer pool to specific LBaaS agent. Pools are scheduled automatically when created. This feature is implemented on top of Agent Management extension. The Load Balancer Agent Scheduler extension is added in Havana.

Verb	URI	Description
GET	/agents/*agent_id*/ loadbalancer-pools	List pools the given LBaaS agent is hosting.
GET	/lb/pools/*pool_id*/ loadbalancer-agent	Show a LBaaS agent hosting the given pool.

List Pools hosted by one LBaaS agent

Verb	URI	Description
GET	/agents/*agent_id*/ loadbalancer-pools	List pools the given LBaaS agent is hosting.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Forbidden (403)

This operation does not require a request body.

This operation returns a response body.

Example 4.160. List Pools on LBaaS agent: JSON Request

```
GET /v2.0/agents/6ee1df7f-bae4-4ee9-910a-d33b000773b0/loadbalancer-pools.json
 HTTP/1.1
Host: localhost:9696
User-Agent: python-neutronclient
Content-Type: application/json
Accept: application/json
X-Auth-Token: 6eeea6e73b68415f85d8368902a32c11
```

Example 4.161. List Pools on LBaaS agent: JSON Response


```
{
    "pools": [
        {
            "admin_state_up": true,
            "description": "",
            "health_monitors": [],
            "health_monitors_status": [],
            "id": "28296abb-e675-4288-9cd0-6c112c720db0",
            "lb_method": "ROUND_ROBIN",
            "members": [],
            "name": "pool1",
            "protocol": "HTTP",
            "provider": "haproxy",
            "status": "PENDING_CREATE",
            "status_description": null,
            "subnet_id": "f8fd83d3-2080-4ab9-9814-391fe7b8a7a4",
            "tenant_id": "54d7b6253c8c4e64862fbd08b3fc08cd",
            "vip_id": null
        }
    ]
}
```


Show LBaaS agent hosting pool

Verb	URI	Description
GET	/lb/pools/*pool_id*/loadbalancer-agent	Show a LBaaS agent hosting the given pool.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Forbidden (403)

This operation does not require a request body.

This operation returns a response body.

Example 4.162. Show LBaaS agent hosting pool: JSON Request

```
GET /v2.0/lb/pools/28296abb-e675-4288-9cd0-6c112c720db0/loadbalancer-agent.
json HTTP/1.1
Host: localhost:9696
User-Agent: python-neutronclient
Content-Type: application/json
Accept: application/json
X-Auth-Token: bce63afb1e794c70972a19a7c2d6dcab
```

Example 4.163. Show LBaaS agent hosting pool: JSON Response


```
{
    "agent": {
        "admin_state_up": true,
        "agent_type": "Loadbalancer agent",
        "alive": true,
        "binary": "neutron-loadbalancer-agent",
        "configurations": {
```

```
            "device_driver": "neutron.services.loadbalancer.drivers.haproxy.
namespace_driver.HaproxyNSDriver",
            "devices": 0,
            "interface_driver": "neutron.agent.linux.interface.
OVSInterfaceDriver"
        },
        "created_at": "2013-10-01 12:50:13",
        "description": null,
        "heartbeat_timestamp": "2013-10-01 12:56:29",
        "host": "ostack02",
        "id": "6ee1df7f-bae4-4ee9-910a-d33b000773b0",
        "started_at": "2013-10-01 12:50:13",
        "topic": "lbaas_process_on_host_agent"
    }
}
```

The Virtual Private Network as a Service (VPNaaS) Extension

The VPNaaS extension provides OpenStack tenants with the ability to extend private networks across the public telecommunication infrastructure. The capabilities provided by this initial implementation of the VPNaaS extension are:

- Site-to-site Virtual Private Network connecting two private networks.
- Multiple VPN connections per tenant.
- Supporting IKEv1 policy with 3des, aes-128, aes-256, or aes-192 encryption.
- Supporting IPSec policy with 3des, aes-128, aes-256, or aes-192 encryption, sha1 authentication, ESP, AH, or AH-ESP transform protocol, and tunnel or transport mode encapsulation.
- Dead Peer Detection (DPD) allowing hold, clear, restart, disabled, or restart-by-peer actions.

This extension introduces new resources:

- **service**, a high level object that associates VPN with a specific subnet and router.
- **ikepolicy**, the Internet Key Exchange policy identifying the authentication and encryption algorithm used during phase one and phase two negotiation of a VPN connection.
- **ipsecpolicy**, the IP security policy specifying the authentication and encryption algorithm, and encapsulation mode used for the established VPN connection.
- **ipsec-site-connection**, has details for the site-to-site IPsec connection, including the peer CIDRs, MTU, authentication mode, peer address, DPD settings, and status.

Note

This extension is **experimental** for the Havana release. The API may change without backward compatibility.

Concepts

A VPN **service** relates the Virtual Private Network with a specific subnet and router for a tenant.

An **IKE Policy** is used for phase one and phase two negotiation of the VPN connection. Configuration selects the authentication and encryption algorithm used to establish a connection.

An **IPsec Policy** is used to specify the encryption algorithm, transform protocol, and mode (tunnel/transport) for the VPN connection.

A VPN **connection** represents the IPsec tunnel established between two sites for the tenant. This contains configuration settings specifying the policies used, peer information, MTU, and the DPD actions to take.

High-level flow

The high-level task flow for using VPNaaS API to configure a site-to-site Virtual Private Network is as follows:

1. The tenant creates a VPN service specifying the router and subnet.
2. The tenant creates an IKE Policy.
3. The tenant creates an IPsec Policy.
4. The tenant creates a VPN connection, specifying the VPN service, peer information, and IKE and IPsec policies.

VPN Service Operations

This section discusses operations for managing a tenant's VPN service through this extension.

Table 4.13. VPN Service Attributes

Attribute	Type	Required	CRUD [a]	Default Value	Validation Constraints	Notes
id	uuid-str	N/A	R	generated	N/A	Unique identifier for the VPN Service object.
tenant_id	uuid-str	Yes	CR	Derived from Authentication token	valid tenant_id	Owner of the VPN service. Only admin users can specify a tenant identifier other than their own.
name	String	No	CRU	None	N/A	Human readable name for the VPN service. Does not have to be unique.
description	String	No	CRU	None	N/A	Human readable description for the VPN service.
status	String	N/A	R	N/A	N/A	Indicates whether IPsec VPN service is currently operational. Possible values include: ACTIVE, DOWN, BUILD, ERROR, PENDING_CREATE, PENDING_UPDATE, or PENDING_DELETE.
admin_state_up	Bool	N/A	CRU	true	{true | false }	Administrative state of the vpnservice. If false (down), port does not forward packets.
subnet_id	uuid-str	Yes	CR	N/A	valid subnet ID	The subnet on which the tenant wants the VPN service. This may be extended in the future to support multiple subnets.
router_id	uuid-str	Yes	CR	N/A	valid router ID	Router ID to which the VPN service is inserted. This may change in the future, when router level insertion is available.

[a]

- **C**. Use the attribute in create operations.
- **R**. This attribute is returned in response to show and list operations.
- **U**. You can update the value of this attribute.
- **D**. You can delete the value of this attribute.

List VPN Services

Verb	URI	Description
GET	/vpn/vpnservices	Lists VPN services.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Forbidden (403)

This operation does not require a request body.

This operation returns a response body.

Example 4.164. List VPN Services: Request

```
GET /v2.0/vpn/vpnservices.json
User-Agent: python-neutronclient
Accept: application/json
```

Example 4.165. List VPN Services: Response

```
{
  "vpnservices": [
    {
      "router_id": "ec8619be-0ba8-4955-8835-3b49ddb76f89",
      "status": "PENDING_CREATE",
      "name": "myservice",
      "admin_state_up": true,
      "subnet_id": "f4fb4528-ed93-467c-a57b-11c7ea9f963e",
      "tenant_id": "ccb81365fe36411a9011e90491fe1330",
      "id": "9faaf49f-dd89-4e39-a8c6-101839aa49bc",
      "description": ""
    }
  ]
}
```

Show VPN Service

Verb	URI	Description
GET	/vpn/vpnservices/*service-id*	Returns details about a specific VPN service.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Forbidden (403), Not Found (404)

This operation does not require a request body.

This operation returns a response body.

Example 4.166. Show VPN Service: Request

```
GET /v2.0/vpn/vpnservices/9faaf49f-dd89-4e39-a8c6-101839aa49bc.json
User-Agent: python-neutronclient
Accept: application/json
```

Example 4.167. Show VPN Service: Response

```
{
  "vpnservice": {
    "router_id": "ec8619be-0ba8-4955-8835-3b49ddb76f89",
    "status": "PENDING_CREATE",
    "name": "myservice",
    "admin_state_up": true,
    "subnet_id": "f4fb4528-ed93-467c-a57b-11c7ea9f963e",
    "tenant_id": "ccb81365fe36411a9011e90491fe1330",
    "id": "9faaf49f-dd89-4e39-a8c6-101839aa49bc",
    "description": ""
  }
}
```

Create VPN Service

Verb	URI	Description
POST	/vpn/vpnservices	Creates a new VPN service.

Normal Response Code: 201

Error Response Codes: Unauthorized (401), Bad Request (400)

This operation requires a request body.

This operation returns a response body.

Example 4.168. Create VPN Service: Request

```
POST /v2.0/vpn/vpnservices.json
User-Agent: python-neutronclient
Accept: application/json
```

```
{
  "vpnservice": {
    "subnet_id": "f4fb4528-ed93-467c-a57b-11c7ea9f963e",
    "router_id": "ec8619be-0ba8-4955-8835-3b49ddb76f89",
    "name": "myservice",
    "admin_state_up": true
  }
}
```

Example 4.169. Create VPN: Response

```
HTTP/1.1 201 Created
Content-Type: application/json; charset=UTF-8
```

```
{
  "vpnservice": {
    "router_id": "ec8619be-0ba8-4955-8835-3b49ddb76f89",
    "status": "PENDING_CREATE",
    "name": "myservice",
    "admin_state_up": true,
    "subnet_id": "f4fb4528-ed93-467c-a57b-11c7ea9f963e",
    "tenant_id": "ccb81365fe36411a9011e90491fe1330",
    "id": "9faaf49f-dd89-4e39-a8c6-101839aa49bc",
    "description": ""
  }
}
```

Update VPN Service

Verb	URI	Description
PUT	/vpn/vpnservices/*service-id*	Updates a VPN service, provided status is not indicating a PENDING_* state.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Bad Request (400), Not Found (404)

Example 4.170. Update VPN Service: Request

```
PUT /v2.0/vpn/vpnservices/41bfef97-af4e-4f6b-a5d3-4678859d2485.json
User-Agent: python-neutronclient
Accept: application/json
```

```
{
  "vpnservice": {
    "description": "Updated description"
  }
}
```

Example 4.171. Update VPN Service: Response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
```

```
{
  "vpnservice": {
    "router_id": "881b7b30-4efb-407e-a162-5630a7af3595",
    "status": "ACTIVE",
    "name": "myvpn",
    "admin_state_up": true,
    "subnet_id": "25f8a35c-82d5-4f55-a45b-6965936b33f6",
    "tenant_id": "26de9cd6cae94c8cb9f79d660d628e1f",
    "id": "41bfef97-af4e-4f6b-a5d3-4678859d2485",
    "description": "Updated description"
  }
}
```

Delete VPN Service

Verb	URI	Description
DELETE	/vpn/vpnservices/*service-id*	Removes a VPN service.

Normal Response Code: 204

Error Response Codes: Unauthorized (401), Not Found (404), Conflict (409)

This operation does not require a request body.

This operation does not return a response body.

Example 4.172. Delete VPN Service: Request

```
DELETE /v2.0/vpn/vpnservices/1be5e5f7-c45e-49ba-85da-156575b60d50.json
User-Agent: python-neutronclient
Accept: application/json
```

Example 4.173. Delete VPN Service: Response

```
HTTP/1.1 204 No Content
Content-Length: 0
```

IKE Policy Operations

This section discusses operations for managing IKE Policies through the VPN as a Service extension.

Table 4.14. IKE Policy Attributes

Attribute	Type	Required	CRUD [a]	Default Value	Validation Constraints	Notes
id	uuid-str	N/A	R	generated	N/A	Unique identifier for the IKE policy.
tenant_id	uuid-str	Yes	CR	None	valid tenant_id	Unique identifier for owner of the VPN service.
name	string	yes	CRU	None	N/A	Friendly name for the IKE policy.
description	string	no	CRU	None	N/A	Description of the IKE policy.
auth_algorithm	string	no	CRU	sha1	N/A	Authentication Hash algorithms: sha1.
encryption_algorithm	string	no	CRU	aes-128	N/A	Encryption Algorithms: 3des, aes-128, aes-256, aes-192, etc.
phase1_negotiation_mode	string	no	CRU	Main Mode	N/A	IKE mode: Main Mode.
pfs	string	no	CRU	Group5	N/A	Perfect Forward Secrecy: Group2, Group5, or Group14.
ike_version	string	no	CRU	v1	N/A	Version: v1 or v2.
lifetime	dict	no	CRU	units: seconds, value: 3600.	Dictionary should be in this form: {'units': 'seconds', 'value': 2000}. Value is a positive integer.	Lifetime of the SA. Units in 'seconds'. Either units or value may be omitted.

[a]

- **C**. Use the attribute in create operations.
- **R**. This attribute is returned in response to show and list operations.
- **U**. You can update the value of this attribute.
- **D**. You can delete the value of this attribute.

List IKE Policies

Verb	URI	Description
GET	/vpn/ikepolicies	Lists IKE policies.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Forbidden (403)

This operation does not require a request body.

This operation returns a response body.

Example 4.174. List IKE Policies: Request

```
GET /v2.0/vpn/ikepolicies.json
User-Agent: python-neutronclient
Accept: application/json
```

Example 4.175. List IKE Policies: Response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
```

```
{
  "ikepolicies": [
    {
      "name": "ikepolicy1",
      "tenant_id": "ccb81365fe36411a9011e90491fe1330",
      "auth_algorithm": "sha1",
      "encryption_algorithm": "aes-256",
      "pfs": "group5",
      "phase1_negotiation_mode": "main",
      "lifetime": {
        "units": "seconds",
        "value": 3600
      },
      "ike_version": "v1",
      "id": "5522aff7-1b3c-48dd-9c3c-b50f016b73db",
      "description": ""
    }
  ]
}
```

Show IKE Policy

Verb	URI	Description
GET	/vpn/ikepolicies/*ikepolicy-id*	Returns details about a specific IKE policy.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Forbidden (403), Not Found (404)

This operation does not require a request body.

This operation returns a response body.

Example 4.176. Show IKE Policy: Request

```
GET /v2.0/vpn/ikepolicies/5522aff7-1b3c-48dd-9c3c-b50f016b73db.json
User-Agent: python-neutronclient
Accept: application/json
```

Example 4.177. Show IKE Policy: Response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
```

```
{
  "ikepolicy": {
    "name": "ikepolicy1",
    "tenant_id": "ccb81365fe36411a9011e90491fe1330",
    "auth_algorithm": "sha1",
    "encryption_algorithm": "aes-256",
    "pfs": "group5",
    "phase1_negotiation_mode": "main",
    "lifetime": {
      "units": "seconds",
      "value": 3600
    },
    "ike_version": "v1",
    "id": "5522aff7-1b3c-48dd-9c3c-b50f016b73db",
    "description": ""
  }
}
```

Create IKE Policy

Verb	URI	Description
POST	/vpn/ikepolicies	Creates a new IKE policy.

Normal Response Code: 201

Error Response Codes: Unauthorized (401), Bad Request (400)

This operation requires a request body.

This operation returns a response body.

Example 4.178. Create IKE Policy: Request

```
POST /v2.0/vpn/ikepolicies.json
User-Agent: python-neutronclient
Accept: application/json
```

```
{
  "ikepolicy": {
    "phase1_negotiation_mode": "main",
    "auth_algorithm": "sha1",
    "encryption_algorithm": "aes-128",
    "pfs": "group5",
    "lifetime": {
      "units": "seconds",
      "value": 7200
    },
    "ike_version": "v1",
    "name": "ikepolicy1"
  }
}
```

Example 4.179. Create IKE Policy: Response

```
HTTP/1.1 201 Created
Content-Type: application/json; charset=UTF-8
```

```
{
  "ikepolicy": {
    "name": "ikepolicy1",
    "tenant_id": "ccb81365fe36411a9011e90491fe1330",
    "auth_algorithm": "sha1",
    "encryption_algorithm": "aes-128",
    "pfs": "group5",
    "phase1_negotiation_mode": "main",
    "lifetime": {
      "units": "seconds",
      "value": 7200
    },
    "ike_version": "v1",
    "id": "5522aff7-1b3c-48dd-9c3c-b50f016b73db",
    "description": ""
```

```
  }
}
```

Update IKE Policy

Verb	URI	Description
PUT	/vpn/ikepolicies/*ikepolicy-id*	Updates an IKE policy.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Bad Request (400), Not Found (404)

Example 4.180. Update IKE Policy: Request

```
PUT /v2.0/vpn/ikepolicies/5522aff7-1b3c-48dd-9c3c-b50f016b73db.json
User-Agent: python-neutronclient
Accept: application/json
```

```
{
  "ikepolicy": {
    "encryption_algorithm": "aes-256"
  }
}
```

Example 4.181. Update IKE Policy: Response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
```

```
{
  "ikepolicy": {
    "name": "ikepolicy1",
    "tenant_id": "ccb81365fe36411a9011e90491fe1330",
    "auth_algorithm": "sha1",
    "encryption_algorithm": "aes-256",
    "pfs": "group5",
    "phase1_negotiation_mode": "main",
    "lifetime": {
      "units": "seconds",
      "value": 3600
    },
    "ike_version": "v1",
    "id": "5522aff7-1b3c-48dd-9c3c-b50f016b73db",
    "description": ""
  }
}
```

Delete IKE Policy

Verb	URI	Description
DELETE	/vpn/ikepolicies/*ikepolicy-id*	Removes an IKE policy.

Normal Response Code: 204

Error Response Codes: Unauthorized (401), Not Found (404), Conflict (409)

This operation does not require a request body.

This operation does not return a response body.

Example 4.182. Delete IKE Policy: Request

```
DELETE /v2.0/vpn/ikepolicies/5522aff7-1b3c-48dd-9c3c-b50f016b73db.json
User-Agent: python-neutronclient
Accept: application/json
```

Example 4.183. Delete IKE Policy: Response

```
HTTP/1.1 204 No Content
Content-Length: 0
```

IPSec Policy Operations

This section discusses operations for managing IPSec policies through the VPN as a Service extension.

Table 4.15. IPSec Policy Attributes

Attribute	Type	Required	CRUD [a]	Default Value	Validation Constraints	Notes
id	uuid-str	N/A	R	generated	N/A	Unique identifier for the IPsec policy.
tenant_id	uuid-str	Yes	CR	None	valid tenant_id	Unique identifier for owner of the VPN service.
name	string	yes	CRU	None	N/A	Friendly name for the IPsec policy.
description	string	no	CRU	None	N/A	Description of the IPSec policy.
transform_protocol	string	no	CRU	ESP	N/A	Transform protocol used: ESP, AH, or AH-ESP.
encapsulation_mode	string	no	CRU	tunnel	N/A	Encapsulation mode: tunnel or transport.
auth_algorithm	string	no	CRU	sha1	N/A	Authentication algorithm: sha1.
encryption_algorithm	string	no	CRU	aes-128	N/A	Encryption Algorithms: 3des, aes-128, aes-256, or aes-192.
pfs	string	no	CRU	group5	N/A	Perfect Forward Secrecy: group2, group5, or group14.
lifetime	dict	no	CRU	units: seconds, value: 3600.	Dictionary should be in this form: {'units': 'seconds', 'value': 2000}. Value is a positive integer.	Lifetime of the SA. Units in 'seconds'. Either units or value may be omitted.

[a]

- **C**. Use the attribute in create operations.
- **R**. This attribute is returned in response to show and list operations.
- **U**. You can update the value of this attribute.
- **D**. You can delete the value of this attribute.

List IPSec Policy

Verb	URI	Description
GET	/vpn/ipsecpolicies	Lists IPSec policies.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Forbidden (403)

This operation does not require a request body.

This operation returns a response body.

Example 4.184. List IPSec Policies: Request

```
GET /v2.0/vpn/ipsecpolicies.json
User-Agent: python-neutronclient
Accept: application/json
```

Example 4.185. List IPSec Policies: Response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
```

```
{
  "ipsecpolicies": [
    {
      "name": "ipsecpolicy1",
      "transform_protocol": "esp",
      "auth_algorithm": "sha1",
      "encapsulation_mode": "tunnel",
      "encryption_algorithm": "aes-128",
      "pfs": "group14",
      "tenant_id": "ccb81365fe36411a9011e90491fe1330",
      "lifetime": {
        "units": "seconds",
        "value": 3600
      },
      "id": "5291b189-fd84-46e5-84bd-78f40c05d69c",
      "description": ""
    }
  ]
}
```

Show IPSec Policy

Verb	URI	Description
GET	/vpn/ipsecpolicies/*ipsecpolicy-id*	Returns details about a specific IPSec policy.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Forbidden (403), Not Found (404)

This operation does not require a request body.

This operation returns a response body.

Example 4.186. Show IPSec Policy: Request

```
GET /v2.0/vpn/ipsecpolicies/5291b189-fd84-46e5-84bd-78f40c05d69c.json
User-Agent: python-neutronclient
Accept: application/json
```

Example 4.187. Show IPSec Policy: Response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
```

```
{
  "ipsecpolicy": {
    "name": "ipsecpolicy1",
    "transform_protocol": "esp",
    "auth_algorithm": "sha1",
    "encapsulation_mode": "tunnel",
    "encryption_algorithm": "aes-128",
    "pfs": "group14",
    "tenant_id": "ccb81365fe36411a9011e90491fe1330",
    "lifetime": {
      "units": "seconds",
      "value": 3600
    },
    "id": "5291b189-fd84-46e5-84bd-78f40c05d69c",
    "description": ""
  }
}
```

Create IPSec Policy

Verb	URI	Description
POST	/vpn/ipsecpolicies	Creates a new IPSec policy.

Normal Response Code: 201

Error Response Codes: Unauthorized (401), Bad Request (400)

This operation requires a request body.

This operation returns a response body.

Example 4.188. Create IPSec Policy: Request

```
POST /v2.0/vpn/ipsecpolicies.json
User-Agent: python-neutronclient
Accept: application/json
```

```
{
  "ipsecpolicy": {
    "name": "ipsecpolicy1",
    "transform_protocol": "esp",
    "auth_algorithm": "sha1",
    "encapsulation_mode": "tunnel",
    "encryption_algorithm": "aes-128",
    "pfs": "group5",
    "lifetime": {
      "units": "seconds",
      "value": 7200
    }
  }
}
```

Example 4.189. Create IPSec Policy: Response

```
HTTP/1.1 201 Created
Content-Type: application/json; charset=UTF-8
```

```
{
  "ipsecpolicy": {
    "name": "ipsecpolicy1",
    "transform_protocol": "esp",
    "auth_algorithm": "sha1",
    "encapsulation_mode": "tunnel",
    "encryption_algorithm": "aes-128",
    "pfs": "group5",
    "tenant_id": "ccb81365fe36411a9011e90491fe1330",
    "lifetime": {
      "units": "seconds",
      "value": 7200
    },
    "id": "5291b189-fd84-46e5-84bd-78f40c05d69c",
    "description": ""
```

```
  }
}
```

Update IPSec Policy

Verb	URI	Description
PUT	/vpn/ipsecpolicies/*ipsecpolicy-id*	Updates a IPSec policy.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Bad Request (400), Not Found (404)

Example 4.190. Update IPSec Policy: Request

```
PUT /v2.0/vpn/ipsecpolicies/5291b189-fd84-46e5-84bd-78f40c05d69c.json
User-Agent: python-neutronclient
Accept: application/json
```

```
{
  "ipsecpolicy": {
    "pfs": "group14"
  }
}
```

Example 4.191. Update IPSec Policy: Response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
```

```
{
  "ipsecpolicy": {
    "name": "ipsecpolicy1",
    "transform_protocol": "esp",
    "auth_algorithm": "sha1",
    "encapsulation_mode": "tunnel",
    "encryption_algorithm": "aes-128",
    "pfs": "group14",
    "tenant_id": "ccb81365fe36411a9011e90491fe1330",
    "lifetime": {
      "units": "seconds",
      "value": 3600
    },
    "id": "5291b189-fd84-46e5-84bd-78f40c05d69c",
    "description": ""
  }
}
```

Delete IPSec Policy

Verb	URI	Description
DELETE	/vpn/ipsecpolicies/*ipsecpolicy-id*	Removes a IPSec policy.

Normal Response Code: 204

Error Response Codes: Unauthorized (401), Not Found (404), Conflict (409)

This operation does not require a request body.

This operation does not return a response body.

Example 4.192. Delete IPSec Policy: Request

```
DELETE /v2.0/vpn/ipsecpolicies/5291b189-fd84-46e5-84bd-78f40c05d69c.json
User-Agent: python-neutronclient
Accept: application/json
```

Example 4.193. Delete IPSec Policy: Response

```
HTTP/1.1 204 No Content
Content-Length: 0
```

IPSec Site Connection Operations

This section discusses operations for managing IPSec site-to-site connections through the VPN as a Service extension.

Table 4.16. IPSec Site Connection Attributes

Attribute	Type	Required	CRUD [a]	Default Value	Validation Constraints	Notes
id	uuid-str	N/A	R	generated	N/A	Unique identifier for the IPSec site-to-site connection.
tenant_id	uuid-str	Yes	CR	None	valid tenant_id	Unique identifier for owner of the VPN service.
name	string	no	CRU	None	N/A	Name for IPSec site-to-site connection.
description	string	no	CRU	None	N/A	Description of the IPSec site-to-site connection.
peer_address	string	yes	CRU	N/A	N/A	Peer gateway public IPv4/ IPv6 address or FQDN.
peer_id	string	yes	CRU	N/A	N/A	Peer router identity for authentication. Can be IPv4/ IPv6 address, e-mail address, key id, or FQDN.
peer_cidrs	list[string]	yes	CRU	N/A	unique list of valid cidr in the form <net_address>/ <prefix>	Peer private CIDRs.
route_mode	string	no	R	static	static	Route mode: static. This will be extended in the future.
mtu	integer	no	CRU	1500	Integer. Minimum is 68 for IPv4 and 1280 for IPv6.	Maximum Transmission Unit to address fragmentation.
auth_mode	string	no	R	psk	psk/certs	Authentication mode: PSK or certificate.
psk	string	yes	CRU	N/A	NO	Pre Shared Key: any string.
initiator	string	no	CRU	bi-directional	bi-directional / response-only	Whether this VPN can only respond to connections or can initiate as well.
admin_state_up	bool	N/A	CRU	TRUE	true / false	Administrative state of VPN connection. If false (down), VPN connection does not forward packets.
status	string	N/A	R	N/A	N/A	Indicates whether VPN connection is currently operational. Possible values include: ACTIVE, DOWN, BUILD, ERROR, PENDING_CREATE, PENDING_UPDATE, or PENDING_DELETE.
ikepolicy_id	uuid	yes	CR	N/A	Unique identifier of IKE policy	Unique identifier of IKE policy.

Attribute	Type	Required	CRUD [a]	Default Value	Validation Constraints	Notes
ipsecpolicy_id	uuid	yes	CR	N/A	Unique identifier of IPSec policy	Unique identifier of IPSec policy.
vpnservice_id	uuid	yes	CR	N/A	Unique identifier of VPN service	Unique identifier of VPN service.
dpd	dict	no	CRU	action: hold, interval: 30, timeout: 120	Dictionary should be in this form: {'action': 'clear', 'interval': 20, 'timeout': 60}. Interval is positive integer. Timeout is greater than interval.	Dead Peer Detection protocol controls. Action: clear, hold, restart, disabled, or restart-by-peer. Interval and timeout in seconds.

[a]

- **C**. Use the attribute in create operations.
- **R**. This attribute is returned in response to show and list operations.
- **U**. You can update the value of this attribute.
- **D**. You can delete the value of this attribute.

List IPSec Site Connections

Verb	URI	Description
GET	/vpn/ipsec-site-connections	Lists the IPSec site-to-site connections.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Forbidden (403)

This operation does not require a request body.

This operation returns a response body.

Example 4.194. List IPSec Site Connections: Request

```
GET /v2.0/vpn/ipsec-site-connections.json
User-Agent: python-neutronclient
Accept: application/json
```

Example 4.195. List IPSec Site Connections: Response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
```

```
{
  "ipsec_site_connections": [
    {
      "status": "PENDING_CREATE",
      "psk": "secret",
      "initiator": "bi-directional",
      "name": "vpnconnection1",
      "admin_state_up": true,
      "tenant_id": "ccb81365fe36411a9011e90491fe1330",
      "description": "",
      "auth_mode": "psk",
      "peer_cidrs": [
        "10.1.0.0/24"
      ],
      "mtu": 1500,
      "ikepolicy_id": "bf5612ac-15fb-460c-9b3d-6453da2fafa2",
      "dpd": {
        "action": "hold",
        "interval": 30,
        "timeout": 120
      },
      "route_mode": "static",
      "vpnservice_id": "c2f3178d-5530-4c4a-89fc-050ecd552636",
      "peer_address": "172.24.4.226",
      "peer_id": "172.24.4.226",
      "id": "cbc152a0-7e93-4f98-9f04-b085a4bf2511",
      "ipsecpolicy_id": "8ba867b2-67eb-4835-bb61-c226804a1584"
    }
  ]
```

```
}
```

Show IPSec Site Connection

Verb	URI	Description
GET	/vpn/ipsec-site-connections/*connection-id*	Returns details about a specific IPSec site-to-site connection.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Forbidden (403), Not Found (404)

This operation does not require a request body.

This operation returns a response body.

Example 4.196. Show IPSec Site Connection: Request

```
GET /v2.0/vpn/ipsec-site-connections/cbc152a0-7e93-4f98-9f04-b085a4bf2511.json
User-Agent: python-neutronclient
Accept: application/json
```

Example 4.197. Show IPSec Site Connection: Response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
```

```
{
  "ipsec_site_connection": {
    "status": "PENDING_CREATE",
    "psk": "secret",
    "initiator": "bi-directional",
    "name": "vpnconnection1",
    "admin_state_up": true,
    "tenant_id": "ccb81365fe36411a9011e90491fe1330",
    "description": "",
    "auth_mode": "psk",
    "peer_cidrs": [
      "10.1.0.0/24"
    ],
    "mtu": 1500,
    "ikepolicy_id": "bf5612ac-15fb-460c-9b3d-6453da2fafa2",
    "dpd": {
      "action": "hold",
      "interval": 30,
      "timeout": 120
    },
    "route_mode": "static",
    "vpnservice_id": "c2f3178d-5530-4c4a-89fc-050ecd552636",
    "peer_address": "172.24.4.226",
    "peer_id": "172.24.4.226",
    "id": "cbc152a0-7e93-4f98-9f04-b085a4bf2511",
    "ipsecpolicy_id": "8ba867b2-67eb-4835-bb61-c226804a1584"
  }
```

```
}
```

Create IPSec Site Connection

Verb	URI	Description
POST	/vpn/ipsec-site-connections	Creates a new IPSec site connection.

Normal Response Code: 201

Error Response Codes: Unauthorized (401), Bad Request (400)

This operation requires a request body.

This operation returns a response body.

Example 4.198. Create IPSec Site Connection: Request

```
POST /v2.0/vpn/ipsec-site-connections.json
User-Agent: python-neutronclient
Accept: application/json
```

```
{
  "ipsec_site_connection": {
    "psk": "secret",
    "initiator": "bi-directional",
    "ipsecpolicy_id": "22b8abdc-e822-45b3-90dd-f2c8512acfa5",
    "admin_state_up": true,
    "peer_cidrs": [
      "10.2.0.0/24"
    ],
    "mtu": "1500",
    "ikepolicy_id": "d3f373dc-0708-4224-b6f8-676adf27dab8",
    "dpd": {
      "action": "disabled",
      "interval": 60,
      "timeout": 240
    },
    "vpnservice_id": "7b347d20-6fa3-4e22-b744-c49ee235ae4f",
    "peer_address": "172.24.4.233",
    "peer_id": "172.24.4.233",
    "name": "vpnconnection1"
  }
}
```

Example 4.199. Create IPSec Site Connection: Response

```
HTTP/1.1 201 Created
Content-Type: application/json; charset=UTF-8
```

```
{
  "ipsec_site_connection": {
    "status": "PENDING_CREATE",
    "psk": "secret",
```

```
    "initiator": "bi-directional",
    "name": "vpnconnection1",
    "admin_state_up": true,
    "tenant_id": "b6887d0b45b54a249b2ce3dee01caa47",
    "description": "",
    "auth_mode": "psk",
    "peer_cidrs": [
      "10.2.0.0/24"
    ],
    "mtu": 1500,
    "ikepolicy_id": "d3f373dc-0708-4224-b6f8-676adf27dab8",
    "dpd": {
      "action": "disabled",
      "interval": 60,
      "timeout": 240
    },
    "route_mode": "static",
    "vpnservice_id": "7b347d20-6fa3-4e22-b744-c49ee235ae4f",
    "peer_address": "172.24.4.233",
    "peer_id": "172.24.4.233",
    "id": "af44dfd7-cf91-4451-be57-cd4fdd96b5dc",
    "ipsecpolicy_id": "22b8abdc-e822-45b3-90dd-f2c8512acfa5"
  }
}
```

Update IPSec Site Connection

Verb	URI	Description
PUT	/vpn/ipsec-site-connections/*connection-id*	Updates an IPSec site-to-site connection, provided status is not indicating a PENDING_* state.

Normal Response Code: 200

Error Response Codes: Unauthorized (401), Bad Request (400), Not Found (404)

Example 4.200. Update IPSec Site Connection: Request

```
PUT /v2.0/vpn/ipsec-site-connections/f7cf7305-f491-45f4-ad9c-8e7240fe3d72.json
User-Agent: python-neutronclient
Accept: application/json
```

```
{
  "ipsec_site_connection": {
    "mtu": "2000"
  }
}
```

Example 4.201. Update IPSec Site Connection: Response

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
```

```
{
```

```
  "ipsec_site_connection": {
    "status": "DOWN",
    "psk": "secret",
    "initiator": "bi-directional",
    "name": "vpnconnection1",
    "admin_state_up": true,
    "tenant_id": "26de9cd6cae94c8cb9f79d660d628e1f",
    "description": "",
    "auth_mode": "psk",
    "peer_cidrs": [
      "10.2.0.0/24"
    ],
    "mtu": 2000,
    "ikepolicy_id": "771f081c-5ec8-4f9a-b041-015dfb7fbbe2",
    "dpd": {
      "action": "hold",
      "interval": 30,
      "timeout": 120
    },
    "route_mode": "static",
    "vpnservice_id": "41bfef97-af4e-4f6b-a5d3-4678859d2485",
    "peer_address": "172.24.4.233",
    "peer_id": "172.24.4.233",
    "id": "f7cf7305-f491-45f4-ad9c-8e7240fe3d72",
    "ipsecpolicy_id": "9958d4fe-3719-4e8c-84e7-9893895b76b4"
  }
}
```

Delete IPSec Site Connection

Verb	URI	Description
DELETE	/vpn/ipsec-site-connections/*connection-id*	Deletes a IPSec site-to-site connection.

Normal Response Code: 204

Error Response Codes: Unauthorized (401), Not Found (404), Conflict (409)

This operation does not require a request body.

This operation does not return a response body.

Example 4.202. Delete IPSec Site Connection: Request

```
DELETE /v2.0/vpn/ipsec-site-connections/cbc152a0-7e93-4f98-9f04-b085a4bf2511.
json
User-Agent: python-neutronclient
Accept: application/json
```

Example 4.203. Delete IPSec Site Connection: Response

```
HTTP/1.1 204 No Content
Content-Length: 0
```

The Allowed Address Pair Extension

The Allowed Address Pair extension extends the port attribute to allow one to specify arbitrary mac_address/ip_address(cidr) pairs that are allowed to pass through a port regardness of the subnet associated with the network.

Port API operations with allowed address pair extension

This section discusses operations for setting and retrieving the allowed address pair extension attributes for port objects.

List Ports

Verb	URI	Description
GET	/ports	Returns a list of ports with their allowed address pair attributes.

Normal Response Code: 200 OK

Error Response Codes: 401 Unauthorized

This operation returns, for each port, its allowed address pair attributes as well as all the attributes normally returned by the list port operation.

Example 4.204. List Ports with allowed address pair attributes: JSON Response

```
{
    "ports":[
    {
     "admin_state_up": true,
     "allowed_address_pairs": [{"ip_address": "23.23.23.1",
                                "mac_address": "fa:16:3e:c4:cd:3f"}],
     "device_id": "",
     "device_owner": "",
     "fixed_ips": [{"ip_address": "10.0.0.2",
                    "subnet_id": "f4145134-b99b-4b18-9940-47239f071923"}],
     "id": "191f5290-3a5a-40ff-b0cb-cd4b115b400e",
     "mac_address": "fa:16:3e:c4:cd:3f",
     "name": "",
     "network_id": "327f2a2f-9d70-417f-ac3a-d3155e25cf25",
     "status": "DOWN",
     "tenant_id": "8462a4d167f84256b7035f4c408c1185"
    },
    {
     "admin_state_up": true,
     "allowed_address_pairs": [],
     "device_id": "",
     "device_owner": "",
     "fixed_ips": [{"ip_address": "10.0.0.3",
                    "subnet_id": "f4145134-b99b-4b18-9940-47239f071923"}],
     "id": "ec2fb9f9-a11b-4791-852d-eb1ab9b27a0e",
     "mac_address": "fa:16:3e:a9:3e:1a",
     "name": "",
     "network_id": "327f2a2f-9d70-417f-ac3a-d3155e25cf25",
     "status": "DOWN",
     "tenant_id": "8462a4d167f84256b7035f4c408c1185"
   }
  ]
}
```

Example 4.205. List Ports with allowed address pair attributes: XML Response

```
<?xml version='1.0' encoding='UTF-8'?>
<ports xmlns="http://openstack.org/quantum/api/v2.0"
```

```
       xmlns:quantum="http://openstack.org/quantum/api/v2.0"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <port>
        <status>ACTIVE</status>
        <name />
        <allowed_address_pairs>
            <allowed_address_pair>
                <ip_address>23.23.23.1</ip_address>
                <mac_address>fa:16:3e:c4:cd:3f</mac_address>
            </allowed_address_pair>
        </allowed_address_pairs>
        <admin_state_up quantum:type="bool">True</admin_state_up>
        <network_id>3537e809-8bec-4ae4-a5ab-2c6477760195</network_id>
        <tenant_id>8462a4d167f84256b7035f4c408c1185</tenant_id>
        <device_owner />
        <mac_address>fa:16:3e:21:4c:2e</mac_address>
        <fixed_ips>
            <fixed_ip>
                <subnet_id>f4145134-b99b-4b18-9940-47239f071923</subnet_id>
                <ip_address>10.0.0.21</ip_address>
            </fixed_ip>
        </fixed_ips>
        <id>191f5290-3a5a-40ff-b0cb-cd4b115b400e</id>
        <device_id />
    </port>
    <port>
        <status>ACTIVE</status>
        <name />
        <allowed_address_pairs xsi:nil="true" />
        <admin_state_up quantum:type="bool">True</admin_state_up>
        <network_id>327f2a2f-9d70-417f-ac3a-d3155e25cf25</network_id>
        <tenant_id>8462a4d167f84256b7035f4c408c1185</tenant_id>
        <device_owner />
        <mac_address>fa:16:3e:a9:3e:1a</mac_address>
        <fixed_ips>
            <fixed_ip>
                <subnet_id>18cf6972-95cc-4134-a986-843dc7433aa0</subnet_id>
                <ip_address>10.0.0.5</ip_address>
            </fixed_ip>
        </fixed_ips>
        <id>ec2fb9f9-a11b-4791-852d-eb1ab9b27a0e</id>
        <device_id />
    </port>
</ports>
```

Show Port

Verb	URI	Description
GET	/ports/*port_id*	Returns details about a specific port, including allowed address pair attributes.

Normal Response Code: 200 OK

Error Response Code: 401 Unauthorized, 404 Not Found

Example 4.206. Show port with allowed address pair attributes: JSON Response

```
{
```

```
   "port":
   {
     "admin_state_up": true,
     "allowed_address_pairs": [{"ip_address": "23.23.23.1",
                                "mac_address": "fa:16:3e:c4:cd:3f"}],
     "device_id": "",
     "device_owner": "",
     "fixed_ips": [{"ip_address": "10.0.0.2",
                    "subnet_id": "f4145134-b99b-4b18-9940-47239f071923"}],
     "id": "191f5290-3a5a-40ff-b0cb-cd4b115b400e",
     "mac_address": "fa:16:3e:c4:cd:3f",
     "name": "",
     "network_id": "327f2a2f-9d70-417f-ac3a-d3155e25cf25",
     "status": "DOWN",
     "tenant_id": "8462a4d167f84256b7035f4c408c1185"
   }
}
```

Example 4.207. Show port with allowed address pair attributes: XML Response

```
<?xml version='1.0' encoding='UTF-8'?>
<port xmlns="http://openstack.org/quantum/api/v2.0"
       xmlns:quantum="http://openstack.org/quantum/api/v2.0"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
    <status>ACTIVE</status>
    <name />
    <allowed_address_pairs>
        <allowed_address_pair>
            <ip_address>23.23.23.1</ip_address>
            <mac_address>fa:16:3e:c4:cd:3f</mac_address>
        </allowed_address_pair>
    </allowed_address_pairs>
    <admin_state_up quantum:type="bool">True</admin_state_up>
    <network_id>3537e809-8bec-4ae4-a5ab-2c6477760195</network_id>
    <tenant_id>8462a4d167f84256b7035f4c408c1185</tenant_id>
    <device_owner />
    <mac_address>fa:16:3e:21:4c:2e</mac_address>
    <fixed_ips>
        <fixed_ip>
            <subnet_id>f4145134-b99b-4b18-9940-47239f071923</subnet_id>
            <ip_address>10.0.0.21</ip_address>
        </fixed_ip>
    </fixed_ips>
    <id>191f5290-3a5a-40ff-b0cb-cd4b115b400e</id>
    <device_id />
</port>
```

Create Port

Verb	URI	Description
POST	/ports	Creates a new port and explicitly specify the allowed address pair attributes.

Normal Response Code: 201

Error Response Code: 400 Bad Request, 401 Unauthorized, 403 Forbidden

Bad request is returned if an allowed address pair matches the mac_address and ip_address on port.

Note: If the mac_address field is left out of the body of the request the mac_address assigned to the port will be used.

Example 4.208. Create Port with allowed address pair attributes: JSON Request

```
{
 "port":
    {
      "network_id": "3537e809-8bec-4ae4-a5ab-2c6477760195",
      "allowed_address_pairs": [{"ip_address": "10.3.3.3"}]
    }
}
```

Update Port

Verb	URI	Description
PUT	/ports/*port_id*	Updates a port, with new allowed address pair values.

Normal Response Code: 200 OK

Error Response Code: 400 Bad Request, 401 Unauthorized, 404 Not Found, 403 Forbidden

Example 4.209. Update allowed address pair attributes for a port: JSON Request

```
{
    "port": {
        "allowed_address_pairs":
         [
            {"ip_address": "10.0.0.1"}
         ]
    }
}
```

The Extra DHCP Options Extension (extra-dhcp-opt)

The Neutron DHCP options extension allows adding DHCP options that are associated to a Neutron Port. They are tagged such that they can be associated from the hosts file to designate a specific network interface and port. The DHCP tag scheme used to associate options to the host files is the port_id (UUID - in the form of 8-4-4-4-12 for a total of 36 characters), these associate options to a Neutron Port and its network. The Dynamic Host Configuration Protocol (DHCP) provides a framework for passing configuration information to hosts on a TCP/IP network. Configuration parameters and other information are carried in tagged data items that are stored in the 'options' field of the DHCP message.

You can specify a DHCP options when defining or updating a Port by specifying the extra_dhcp_opts tag and providing its options as name value pairs e.g. opt_name='bootfile-name', opt_value='pxelinux.0'.

Concepts

The extra-dhcp-opt extension is an attribute extension which adds the following set of attributes to the **port** resource:

- *extra-dhcp-opt:opt_name* - Specified the DHCP option that this is defined as mapped to this port resource. Examples are `bootfile-name`, `server-ip-address`, `tftp-server`, etc..
- *extra-dhcp-opt:opt_value* - Identifies the value associated with the opt_name. These are handled in opt_name, opt_value pairs only. value_opt can be any text string depending upon the name.

The actual semantics of extra-dhcp-opt attributes depend on the name of the dhcp option being used that defines the vendor extension to DHCP. For example reference RFC: http://tools.ietf.org/html/rfc2132, contains specific detail on BOOTP Vendor Extensions.

Port API operations with extra-dhcp-opt port extension

This section discusses operations for setting and retrieving the extra-dhcp-opt port extension attributes for port objects.

List Ports

Verb	URI	Description
GET	/ports	Returns a list of ports with their attributes.

Normal Response Code: 200 OK

Error Response Codes: 401 Unauthorized

This operation returns, all the ports defined in Neutron that this user has access to.

Example 4.210. List Ports with extra_dhcp_opts: JSON Response


```
{
  "ports": [
  {
    "status": "DOWN",
    "binding:host_id": null,
    "name": "",
    "allowed_address_pairs": [],
    "admin_state_up": true,
    "network_id": "87733bcc-8144-41b1-bb6b-d011d7a5168e",
    "tenant_id": "7ea98790cd854fb5a82ef3d41e5c156b",
    "extra_dhcp_opts": [{"opt_value": "testfile.1", "opt_name": "bootfile-name"},
 {"opt_value": "123.123.123.45", "opt_name": "server-ip-address"}, {"opt_value": "123.
123.123.123", "opt_name": "tftp-server"}],
    "binding:vif_type": "ovs",
    "device_owner": "",
    "binding:capabilities": {"port_filter": true},
    "mac_address": "fa:16:3e:52:92:3a",
```

```
    "fixed_ips": [{"subnet_id": "99a8aea3-b9da-409d-a5e5-f45338ceb4d3", "ip_address":
 "172.24.4.228"}],
    "id": "3c0c7a37-690a-43a8-8088-5d4c2c7f8484",
    "security_groups": ["9bf6f19a-ba4a-470f-b8ce-28c9ad66556c"],
    "device_id": ""
  },
  {
    "status": "ACTIVE",
    "binding:host_id": null,
    "name": "",
    "allowed_address_pairs": [],
    "admin_state_up": true,
    "network_id": "87733bcc-8144-41b1-bb6b-d011d7a5168e",
    "tenant_id": "7ea98790cd854fb5a82ef3d41e5c156b",
    "extra_dhcp_opts": [],
    "binding:vif_type": "ovs",
    "device_owner": "compute:probe",
    "binding:capabilities": {"port_filter": true},
    "mac_address": "fa:16:3e:49:56:07",
    "fixed_ips": [{"subnet_id": "99a8aea3-b9da-409d-a5e5-f45338ceb4d3", "ip_address":
 "172.24.4.227"}],
    "id": "5212d40a-c2f5-4a5d-ad18-694658047654",
    "security_groups": ["9bf6f19a-ba4a-470f-b8ce-28c9ad66556c"],
    "device_id": "zvm2"
  }
  ]
}
```

Show Port

Verb	URI	Description
GET	/ports/*port_id*	Returns details about a specific port, including extra-dhcp-opt attributes.

Normal Response Code: 200 OK

Error Response Code: 401 Unauthorized, 404 Not Found

This operation returns, for the port specified in the request URI, its port attributes, including the extra_dhcp_opts attributes.

Example 4.211. Show port with extra-dhcp-opt attributes: JSON Response

```
{
    "port":
    {
        "status": "DOWN",
        "binding:host_id": null,
        "name": "",
        "allowed_address_pairs": [],
        "admin_state_up": true,
        "network_id": "87733bcc-8144-41b1-bb6b-d011d7a5168e",
        "tenant_id": "7ea98790cd854fb5a82ef3d41e5c156b",
        "extra_dhcp_opts": [
            {"opt_value": "testfile.1","opt_name": "bootfile-name"},
            {"opt_value": "123.123.123.123", "opt_name": "tftp-server"},
            {"opt_value": "123.123.123.45", "opt_name": "server-ip-address"}
        ],
        "binding:vif_type": "ovs",
        "device_owner": "",
        "binding:capabilities": {"port_filter": true},
        "mac_address": "fa:16:3e:52:92:3a",
        "fixed_ips": [{"subnet_id": "99a8aea3-b9da-409d-a5e5-f45338ceb4d3",
 "ip_address": "172.24.4.228"}],
```

```
        "id": "3c0c7a37-690a-43a8-8088-5d4c2c7f8484",
        "security_groups": ["9bf6f19a-ba4a-470f-b8ce-28c9ad66556c"],
        "device_id": ""
     }
}
```

Create Port

Verb	URI	Description
POST	/ports	Create a new port and explicitly specify attributes with the extra-dhcp-opt extension attributes.

Normal Response Code: 200 OK

Error Response Code: 401 Unauthorized.

This operation returns, for the port specified in the request URI, its port attributes, including the extra_dhcp_opts attributes.

Example 4.212. Create port with extra-dhcp-opt attributes: JSON Request

```
{
    "port":
    {
        "network_id": "87733bcc-8144-41b1-bb6b-d011d7a5168e",
        "extra_dhcp_opts": [
            {"opt_value": "pxelinux.0", "opt_name": "bootfile-name"},
            {"opt_value": "123.123.123.123", "opt_name": "tftp-server"},
            {"opt_value": "123.123.123.45", "opt_name": "server-ip-address"}
        ],
        "fixed_ips": [{"subnet_id": "99a8aea3-b9da-409d-a5e5-f45338ceb4d3",
 "ip_address": "172.24.4.230"}],
        "admin_state_up": true
    }
}
```

Example 4.213. Create port with extra-dhcp-opt attributes: JSON Response

```
{
    "port":
    {
        "status": "DOWN",
        "binding:host_id": null,
        "name": "",
        "allowed_address_pairs": [],
        "admin_state_up": true,
        "network_id": "87733bcc-8144-41b1-bb6b-d011d7a5168e",
        "tenant_id": "7ea98790cd854fb5a82ef3d41e5c156b",
        "extra_dhcp_opts": [
            {"opt_value": "123.123.123.123", "opt_name": "tftp-server"},
            {"opt_value": "pxelinux.0", "opt_name": "bootfile-name"},
            {"opt_value": "123.123.123.45", "opt_name": "server-ip-address"}
        ],
        "binding:vif_type": "ovs",
        "device_owner": "",
        "binding:capabilities": {"port_filter": true},
        "mac_address": "fa:16:3e:43:3c:b7",
        "fixed_ips": [{"subnet_id": "99a8aea3-b9da-409d-a5e5-f45338ceb4d3",
 "ip_address": "172.24.4.230"}],
        "id": "055d27c0-0194-4782-be45-275ff2c95c61",
        "security_groups": ["9bf6f19a-ba4a-470f-b8ce-28c9ad66556c"],
        "device_id": ""
    }
}
```

Update Port

Verb	URI	Description
PUT	/ports/*port_id*	Updates a port's attributes, including extra_dhcp_opts extension attributes.

Normal Response Code: 200 OK

Error Response Code: 401 Unauthorized.

This operation allow for the updating of attributes for the port specified in the request URI, its port attributes, including the extra_dhcp_opts attributes.

Example 4.214. Update port with extra-dhcp-opt attributes: JSON Request

```
{
    "port":
    {
        "extra_dhcp_opts": [{"opt_value": "testfile.1", "opt_name": "bootfile-name"}]
     }
}
```

Example 4.215. Update port with extra-dhcp-opt attributes: JSON Response

```
{
    "port":
    {
        "status": "DOWN",
        "binding:host_id": null,
        "name": "",
        "allowed_address_pairs": [],
        "admin_state_up": true,
        "network_id": "87733bcc-8144-41b1-bb6b-d011d7a5168e",
        "tenant_id": "7ea98790cd854fb5a82ef3d41e5c156b",
        "extra_dhcp_opts":
        [
            {"opt_value": "123.123.123.123", "opt_name": "tftp-server"},
            {"opt_value": "testfile.1", "opt_name": "bootfile-name"},
            {"opt_value": "123.123.123.45", "opt_name": "server-ip-address"}
        ],
        "binding:vif_type": "ovs",
        "device_owner": "",
        "binding:capabilities": {"port_filter": true},
        "mac_address": "fa:16:3e:43:3c:b7",
        "fixed_ips": [{"subnet_id": "99a8aea3-b9da-409d-a5e5-f45338ceb4d3",
 "ip_address": "172.24.4.230"}],
        "id": "055d27c0-0194-4782-be45-275ff2c95c61",
        "security_groups": ["9bf6f19a-ba4a-470f-b8ce-28c9ad66556c"],
        "device_id": ""
    }
}
```

Metering labels and rules

Creates, modifies, and deletes OpenStack Layer3 Metering labels and rules.

Method	URI	Description
GET	`/metering-labels`	Lists a summary of all l3 metering labels belonging to the specified tenant.

Method	URI	Description
POST	`/metering-labels`	Creates a l3 metering label.
GET	`/metering-labels/{metering_label_id}`	Shows informations for a specified metering label.
DELETE	`/metering-labels/{metering_label_id}`	Deletes a l3 metering label.
GET	`/metering-label-rules`	Lists a summary of all l3 metering label rules belonging to the specified tenant.
POST	`/metering-label-rules`	Creates a l3 metering label rule.
GET	`/metering-label-rules/{metering-label-rule-id}`	Shows detailed informations for a specified metering label rule.
DELETE	`/metering-label-rules/{metering-label-rule-id}`	Deletes a specified l3 metering label rule.

List Metering Labels

Method	URI	Description
GET	/metering-labels	Lists a summary of all l3 metering labels belonging to the specified tenant.

The list includes the unique ID for each metering labels.

This operation does not require a request body.

This operation returns a response body.

Normal response codes: 200

Error response codes: unauthorized (401)

Request

Example 4.216. List Metering Labels: JSON request

```
GET /v2.0/metering/metering-labels HTTP/1.1
Host: controlnode:9696
User-Agent: python-neutronclient
Content-Type: application/json
Accept: application/json
X-Auth-Token: c52a1b304fec4ca0ac85dc1741eec6e2
```

This operation does not require a request body.

Response

Example 4.217. List Metering Labels: JSON response

```
{
  "metering_labels": [
    {
      "tenant_id": "45345b0ee1ea477fac0f541b2cb79cd4",
      "description": "label1 description",
      "name": "label1",
      "id": "a6700594-5b7a-4105-8bfe-723b346ce866"
    },
    {
      "tenant_id": "45345b0ee1ea477fac0f541b2cb79cd4",
      "description": "label2 description",
      "name": "label2",
      "id": "e131d186-b02d-4c0b-83d5-0c0725c4f812"
    }
  ]
}
```

Create Metering Label

Method	URI	Description
POST	/metering-labels	Creates a l3 metering label.

This operation requires a request body.

The following table describes the required and optional attributes in the request body:

Table 4.17. Create Metering Label Rule Request Attributes

Attribute	Required	Description
name	Required	The name of the metering label.
description	Optional	Description for the metering label.

This operation returns a response body, which contains the following informations about the metering label:

- name. Name of the metering label.
- description. Description of the metering label.
- tenant_id. The tenant ID for the specified metering label.
- id. The metering label ID

Normal response codes: 201

Error response codes: badRequest (400), unauthorized (401)

Request

Example 4.218. Create Metering Label: JSON request

```
{
   "metering_label":{
      "name":"label1",
      "description":"description of label1"
   }
}
```

Response

Example 4.219. Create Metering Label: JSON response

```
{
  "metering_label": {
    "tenant_id": "45345b0ee1ea477fac0f541b2cb79cd4",
    "description": "description of label1",
    "name": "label1",
    "id": "bc91b832-8465-40a7-a5d8-ba87de442266"
```

```
  }
}
```

Show Metering Label

Method	URI	Description
GET	/metering-labels/{metering_label_id}	Shows informations for a specified metering label.

This operation does not require a request body.

This operation returns a response body that contains the description, name, ID.

Normal response codes: 200

Error response codes: unauthorized (401), itemNotFound (404)

Request

This table shows the URI parameters for the show metering label request:

Name	Type	Description
{metering_label_id}	Uuid	The unique identifier of the metering label.

Example 4.220. Show Metering Label: JSON request

```
GET /v2.0/metering/metering-labels/a6700594-5b7a-4105-8bfe-723b346ce866 HTTP/
1.1
Host: controlnode:9696
User-Agent: python-neutronclient
Content-Type: application/json
Accept: application/json
X-Auth-Token: c52a1b304fec4ca0ac85dc1741eec6e2
```

This operation does not require a request body.

Response

Example 4.221. Show Metering Label: JSON response

```
{
  "metering_label": {
    "tenant_id": "45345b0ee1ea477fac0f541b2cb79cd4",
    "description": "label1 description",
    "name": "label1",
    "id": "a6700594-5b7a-4105-8bfe-723b346ce866"
  }
}
```

Delete Metering Label

Method	URI	Description
DELETE	`/metering-labels/{metering_label_id}`	Deletes a l3 metering label.

This operation deletes a l3 metering label.

This operation does not require a request body. This operation does not return a response body.

Normal response codes: 204

Error response codes: unauthorized (401), itemNotFound (404)

Request

This table shows the URI parameters for the delete metering label request:

Name	Type	Description
`{metering_label_id}`	Uuid	The unique identifier of the metering label.

Example 4.222. Delete Metering Label: JSON request

```
DELETE /v2.0/metering/metering-labels/a6700594-5b7a-4105-8bfe-723b346ce866
 HTTP/1.1
Host: controlnode:9696
User-Agent: python-neutronclient
Content-Type: application/json
Accept: application/json
X-Auth-Token: c52a1b304fec4ca0ac85dc1741eec6e2
```

This operation does not require a request body.

Response

Example 4.223. Delete Metering Label: JSON response

```
status: 204
```

This operation does not return a response body.

List Metering Label Rules

Method	URI	Description
GET	/metering-label-rules	Lists a summary of all l3 metering label rules belonging to the specified tenant.

The list provides the unique ID for each metering label rule.

This operation does not require a request body. This operation returns a response body.

Normal response codes: 200

Error response codes: unauthorized (401)

Request

Example 4.224. List Metering Label Rules: JSON request

```
GET /v2.0/metering/metering-label-rules HTTP/1.1
Host: controlnode:9696
User-Agent: python-neutronclient
Content-Type: application/json
Accept: application/json
X-Auth-Token: c52a1b304fec4ca0ac85dc1741eec6e2
```

This operation does not require a request body.

Response

Example 4.225. List Metering Label Rules: JSON response

```
{
  "metering_label_rules": [
    {
      "remote_ip_prefix": "20.0.0.0\/24",
      "direction": "ingress",
      "metering_label_id": "e131d186-b02d-4c0b-83d5-0c0725c4f812",
      "id": "9536641a-7d14-4dc5-afaf-93a973ce0eb8",
      "excluded": false
    },
    {
      "remote_ip_prefix": "10.0.0.0\/24",
      "direction": "ingress",
      "metering_label_id": "e131d186-b02d-4c0b-83d5-0c0725c4f812",
      "id": "ffc6fd15-40de-4e7d-b617-34d3f7a93aec",
      "excluded": false
    }
  ]
}
```

Create Metering Label Rule

Method	URI	Description
POST	/metering-label-rules	Creates a l3 metering label rule.

This operation requires a request body.

The following table describes the required and optional attributes in the request body:

Table 4.18. Create Metering Label Rule Request Attributes

Attribute	Required	Description
direction	Optional	Ingress or egress: The direction in which metering rule is applied. Default: ingress
metering_label_id	Required	The meteting label ID to associate with this metering rule.
excluded	Optional	Specify whether the remote_ip_prefix will be excluded or not from traffic counters of the metering label, ie: to not count the traffic of a specific IP address of a range. Default: False
remote_ip_prefix	Required	The remote IP prefix to be associated with this metering rule. packet.

This operation returns a response body.

Normal response codes: 201

Error response codes: badRequest (400), unauthorized (401), itemNotFound (404), buildInProgress (409)

Request

Example 4.226. Create Metering Label Rule: JSON request

```
{
   "metering_label_rule":{
      "remote_ip_prefix":"10.0.1.0/24",
      "direction":"ingress",
      "metering_label_id":"e131d186-b02d-4c0b-83d5-0c0725c4f812"
   }
}
```

Response

Example 4.227. Create Metering Label Rule: JSON response

```
{
```

```
   "metering_label_rule":{
      "remote_ip_prefix":"10.0.1.0/24",
      "direction":"ingress",
      "metering_label_id":"e131d186-b02d-4c0b-83d5-0c0725c4f812",
      "id":"00e13b58-b4f2-4579-9c9c-7ac94615f9ae",
      "excluded":false
   }
}
```

Show Metering Label Rule

Method	URI	Description
GET	`/metering-label-rules/{metering-label-rule-id}`	Shows detailed informations for a specified metering label rule.

This operation does not require a request body.

This operation returns a response body, which contains the following informations about the metering label rule:

- `direction`. Either ingress or egress.
- `excluded`. Either True or False.
- The ID for the specified metering label rule
- The remote IP prefix
- The metering label ID for the metering label with which the rule is associated

Normal response codes: 200

Error response codes: unauthorized (401), itemNotFound (404)

Request

This table shows the URI parameters for the show metering label rule request:

Name	Type	Description
`{metering-label-rule-id}`	Uuid	The unique identifier of metering label rule.

Example 4.228. Show Metering Label Rule: JSON request

```
GET /v2.0/metering/metering-label-rules/9536641a-7d14-4dc5-afaf-93a973ce0eb8
 HTTP/1.1
Host: controlnode:9696
User-Agent: python-neutronclient
Content-Type: application/json
Accept: application/json
X-Auth-Token: c52a1b304fec4ca0ac85dc1741eec6e2
```

This operation does not require a request body.

Response

Example 4.229. Show Metering Label Rule: JSON response

```
{
  "metering_label_rule": {
    "remote_ip_prefix": "20.0.0.0\/24",
    "direction": "ingress",
    "metering_label_id": "e131d186-b02d-4c0b-83d5-0c0725c4f812",
    "id": "9536641a-7d14-4dc5-afaf-93a973ce0eb8",
```

```
    "excluded": false
  }
}
```

Delete Metering Label Rule

Method	URI	Description
DELETE	`/metering-label-rules/{metering-label-rule-id}`	Deletes a specified l3 metering label rule.

This operation does not require a request body.

This operation does not return a response body.

Normal response codes: 204

Error response codes: unauthorized (401), itemNotFound (404)

Request

This table shows the URI parameters for the delete metering label rule request:

Name	Type	Description
`{metering-label-rule-id}`	Uuid	The unique identifier of metering label rule.

Example 4.230. Delete Metering Label Rule: JSON request

```
DELETE /v2.0/metering/metering-labels/37b31179-71ee-4f0a-b130-0eeb28e7ede7
 HTTP/1.1
Host: controlnode:9696
User-Agent: python-neutronclient
Content-Type: application/json
Accept: application/json
X-Auth-Token: c52a1b304fec4ca0ac85dc1741eec6e2
```

This operation does not require a request body.

Response

Example 4.231. Delete Metering Label Rule: JSON response

```
status: 204
```

This operation does not return a response body.