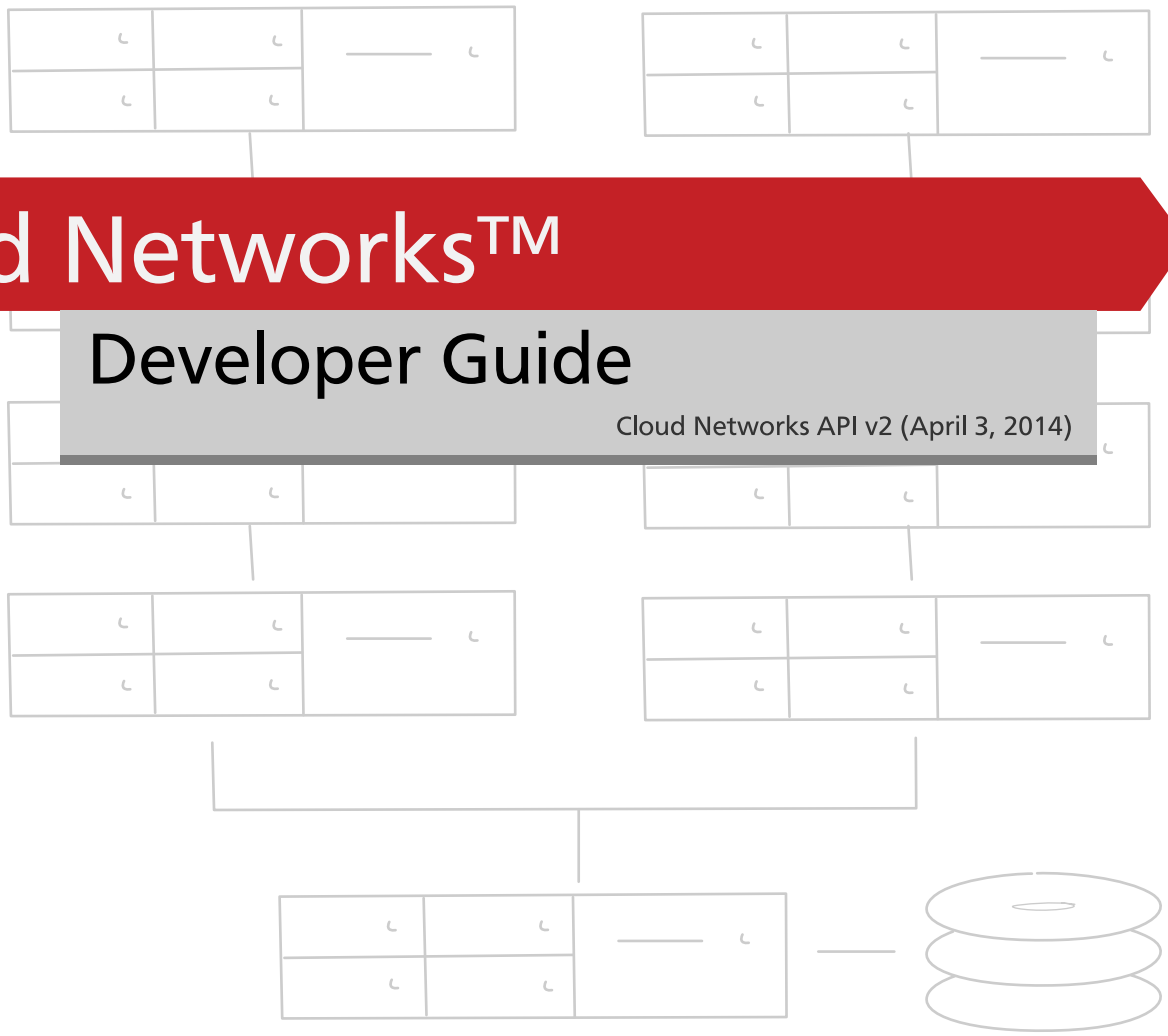
Cloud Networks™

Developer Guide

Cloud Networks API v2 (April 3, 2014)



rackspace®
the open cloud company

docs.rackspace.com/api

Cloud Networks™ Developer Guide

Cloud Networks API v2 (2014-04-03)

This document is intended for software developers who want to develop applications using Rackspace Cloud Networks™, which is powered by the OpenStack Networking code base. In addition to the core features of the OpenStack Networking Application Programming Interface (API) v2, Rackspace has also deployed certain extensions as permitted by the OpenStack Quantum API contract and the OpenStack Neutron API contract. The document is for informational purposes only and is provided "AS IS."

Table of Contents

List of Tables

1. Preface

Rackspace Cloud Networks enables you to create isolated networks and provision *server* instances with Rackspace networks or the isolated networks that you created.

When you create an isolated network, it is associated with your tenant ID.

Cloud Networks was originally released using the Cloud Servers Nova-network (see Chapter 6, "API Operations - Nova-network" [103]). Now Cloud Networks has been released using Neutron (see Chapter 5, "API Operations - Neutron" [28]). Both sets of API operations work. For information that helps you determine which method is better for you, see the section called "Networking: Nova-network versus Neutron" [5].

To request access to Cloud Networks now, click "I want Cloud Networks".

To detect if you are enabled to use Cloud Networks, issue a **GET /limits** call or a **nova absolute-limits** call and review the `maxTotalPrivateNetworks` value:

Table 1.1. maxTotalPrivateNetworks Value

maxTotalPrivateNetworks Value	Description
0	Cloud Networks is disabled.
Greater than 0	Cloud Networks is enabled. The value indicates the number of isolated networks that you are allowed to create.

For more information about querying limits, see Get Limits.

We welcome feedback, comments, and bug reports. Login to the Rackspace customer portal at http://www.rackspace.com/support/.

Intended Audience

This guide assists software developers who want to develop applications by using Cloud Servers and Cloud Networks.

To use this information, you should have access to an active Rackspace Cloud Servers account and access to Cloud Networks. You should also be familiar with the following concepts:

- Cloud Servers service
- *RESTful* web services
- *HTTP*/1.1
- JSON or XML data serialization formats

Pricing and Service Level

Next generation Cloud Networks is part of the Rackspace Cloud and your use through the API is billed according to the pricing schedule for Cloud Servers at http://www.rackspace.com/cloud/servers/pricing/.

The Service Level Agreement (SLA) for Cloud Networks, as part of Cloud Servers, is available at http://www.rackspace.com/cloud/servers/service-levels/.

Document Change History

This version of the guide applies only to v3.0 of the API. The most recent changes are described in the following table:

Revision Date	Summary of Changes
February 14, 2014	Reorganized document and added Neutron API calls.
January 17, 2014	Fixed XML response example for list virtual interfaces.
December 10, 2013	Fixed Identity operation links in the section called "Assigning Roles to Account Users" [19].
December 4, 2013	Updated the name for Managed Cloud Service Level.
October 8, 2013	Added Role Based Access Control [19].
June 24, 2013	Added important note that the customer can choose any endpoint regardless of account origin.
May 23, 2013	Updated "control panel" to "Control Panel."
May 15, 2013	Updated the successful return code for the Delete Virtual Interface API operation.
May 13, 2013	Added the Cloud Networks virtual interface extension, which enables you to create virtual interfaces for, list virtual interfaces for, and delete virtual interfaces from existing server instances.
February 27, 2013	Removed the 421 error code.
December 4, 2012	Corrected the NOVA_RACK_AUTH environment variable to NOVA_RAX_AUTH.
November 13, 2012	Corrected formatting in the PDF.
October 31, 2012	Corrected link to request Cloud Networks access.
October 30, 2012	• Cloud Networks phased-release launch. • Added a preface. • Added JSON and XML request and response examples to the "API operations" chapter.
October 19, 2012	• Added a note about how to attach an isolated network to an existing server. • Added information about programmatically querying limits to determine if you are enabled to use Cloud Networks.
October 17, 2012	Added a note about the error message received by RackConnect and Managed Cloud Service Level customers who opt out of attaching to PublicNet or ServiceNet network when they create a server.
October 12, 2012	Moved Appendix A to the back of this guide.
October 9, 2012	Updated information about ServiceNet.
October 8, 2012	Updated the section about how cURL works.
October 5, 2012	• Updated the authentication topic to describe the fields in the response. • Created a separate *Cloud Networks Developer Guide*. • Added the supernova client installation instructions. • Updated the descriptions of the **nova boot** command parameters.

Resources

Next generation Cloud Servers v2	Cloud Networks v2	Identity v2
• *Next Generation Cloud Servers Release Notes* • *Next Generation Cloud Servers Getting Started* • *Next Generation Cloud Servers Developer Guide*	• *Cloud Networks Release Notes* • *Cloud Networks Getting Started* • *Cloud Networks Developer Guide*	• *Cloud Identity Client Developer Guide v2.0*

For additional Cloud Servers and Cloud Networks service resources, see the Rackspace Cloud site, which provides related documents and links to Rackspace support channels including Knowledge Center articles, phone, chat, and tickets.

For product updates and announcements through Twitter, see http://twitter.com/rackspace.

For information about the supernova client, which is an unsupported wrapper for the nova client useful for managing multiple nova environments, see supernova client.

2. Overview

Cloud Networks enables you to create a virtual Layer 2 network, known as an isolated network, which gives you greater control and security when you deploy web applications.

When you create a next generation Cloud Server, Cloud Networks enables you to attach one or more networks to your server. You can attach an isolated network that you have created or a Rackspace network.

If you install the Cloud Networks virtual interface extension, you can create a virtual interface to a specified Rackspace or isolated network and attach that network to an existing server instance. You can also list virtual interfaces for and delete virtual interfaces from a server instance. For information about the Cloud Networks virtual interface extension, see the section called "Extension - Cloud Networks Virtual Interfaces" [115].

Cloud Networks enables you to attach one or more of the following networks to your server:

- ***PublicNet***. Provides access to the Internet, to Rackspace services (such as Cloud Monitoring, Managed Cloud Service Level support, RackConnect, and Cloud Backup), and to certain operating system updates.

  When you list networks through Cloud Networks, PublicNet is labeled `public`.

- ***ServiceNet***. Provides access to Rackspace services such as Cloud Files, Cloud Databases, and Cloud Backup, and to certain packages and patches through an internal-only, multi-tenant network connection within each Rackspace data center.

  When you list networks through Cloud Networks, ServiceNet is labeled `private`.

  You can use ServiceNet for communications among web servers, application servers, and database servers without incurring bandwidth charges. However, without an isolated network, you must apply security rules to protect data integrity. When you add or remove a server, you must update the security rules on individual servers to permit or deny connections from newly added servers or removed servers.

- **Isolated**. Enables you to deploy web applications on a virtual Layer 2 network that you create through Cloud Networks. An isolated network keeps your server separate from PublicNet, ServiceNet, or both. When you create a isolated network, it is associated with your tenant ID.

When you provision a new server, the networks that are attached to it depend on which of the following methods you use to provision it:

- **The Cloud Servers API**. You must specify the networks that you want to attach to your server. If you do not specify any networks, ServiceNet and PublicNet are attached by default. However, if you specify an isolated network, you must explicitly specify the UUIDs for PublicNet and ServiceNet to attach these networks to your server. The UUID

for ServiceNet is `11111111-1111-1111-1111-111111111111`, and the UUID for PublicNet is `00000000-0000-0000-0000-000000000000`.

- **The nova boot command**. You must specify the networks that you want to attach to your server. If you do not specify any networks, ServiceNet and PublicNet are attached by default. To attach to isolated networks that you have created, you must explicitly specify them in the command. If you do so, those networks, in addition to PublicNet and ServiceNet, are attached to your server.

- **The Cloud Control Panel**. PublicNet and ServiceNet are automatically attached, but you can disable these networks during the server creation process. You can also attach any isolated networks that you have created.

Note

You can explicitly opt out of attaching to the Rackspace networks, which introduces certain potential complications. For more information, see the section called "Limitations of Detaching from Rackspace Networks" [6].

You can use Cloud Networks to perform the following tasks:

- List networks to which the specified tenant has access.
- Create isolated networks.
- Show details for isolated networks.
- Delete an isolated network, but only if it is not associated with any server.

  To detach a network from a server, you must use the Cloud Networks virtual interface extension to delete the virtual interface for the network from the server. See the section called "Extension - Cloud Networks Virtual Interfaces" [115].

- Manage subnets.
- Manage ports.
- Manage Layer-3 routers.

To list the networks that are attached to servers, issue a Cloud Servers List Servers operation. For more information, see List Servers in the *Cloud Servers Developer Guide*.

Networking: Nova-network versus Neutron

Rackspace first introduced networking services by using the Nova-network API based on OpenStack. Now we are providing the Neutron API, also based on OpenStack, which offers more functionality and flexibility than the Nova-network API. Both sets of APIs continue to work well, but the Neutron API will be the base for all the future networking services that Rackspace offers.

The Neutron API provides three primary, top-level resources (networks, ports, and subnets) and includes the ability to perform the following functions:

- Use all existing create, read, update, delete (CRUD) API operations for networks
- Attach and detach networks
- Self-service additional IP addresses
- Pass custom routes to servers
- Set the default gateway on cloud networks
- View quotas related to networks

Note

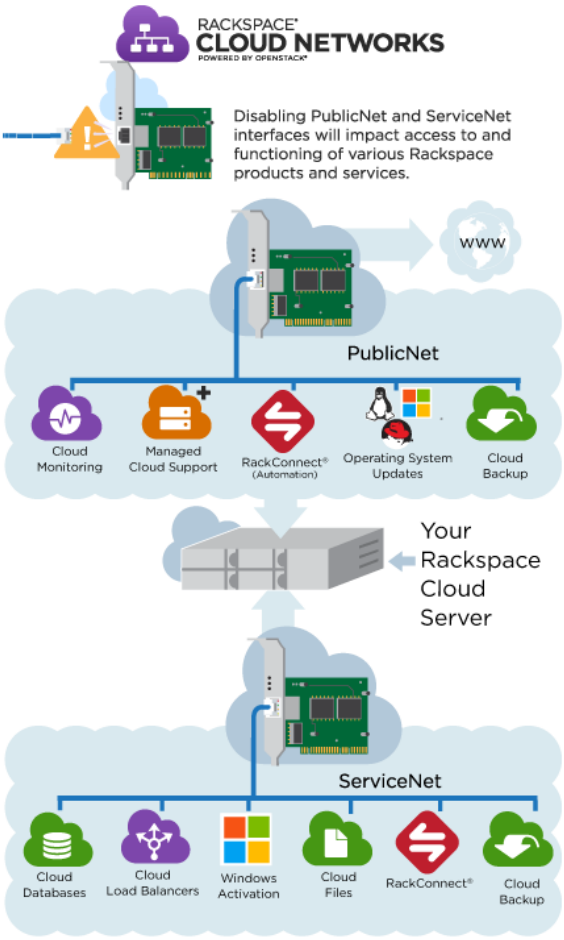
The Neutron API operations return only JSON-formatted responses. There is no option to receive responses in XML.

Limitations of Detaching from Rackspace Networks

When you provision a next generation Cloud Server, you can attach to one or more isolated networks and the Rackspace networks. You can also explicitly opt out of attaching the Rackspace networks to it, which introduces the following limitations:

- If you do not attach the PublicNet network to your server, the server cannot access to the Internet and some Rackspace products and services.
- If you do not attach the ServiceNet network to your server, the server cannot access certain Rackspace products and services.

The following graphic depicts the services that are available only when you attach the Rackspace networks to your server:



To opt out of attaching the Rackspace networks to your server, perform the following action depending on how you are provisioning the server:

- **nova boot command**. Specify the optional `--no-public` and `--no-service-net` parameters.
- **Cloud Servers API**. If you do not specify any networks, ServiceNet and PublicNet are attached by default. However, if you specify an isolated network, you must explicitly specify the UUIDs for PublicNet and ServiceNet to attach these networks to your server.

  The UUID for ServiceNet is `11111111-1111-1111-1111-111111111111`, and the UUID for PublicNet is `00000000-0000-0000-0000-000000000000`.

  Omit these UUIDs from the request to detach from these networks.
- **Cloud Control Panel**. Clear either or both of the PublicNet and ServiceNet selections during the server creation process. You are warned that your capabilities might be degraded by this choice.

3. Concepts

Use the Cloud Networks 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" [8].

Subnet — An IPv4 or IPv6 address block from which IP addresses that are assigned to servers on a specified network are selected. See the section called "Subnet" [9].

Ports — A virtual, or logical, switch port on a specified network. See the section called "Port" [10].

Router — A router is used with a layer-3 extension to interconnect subnets and forward traffic among them. See the section called "Router" [11].

These entities have autogenerated unique identifiers and support basic create, read, update, and delete (CRUD) functions with the POST, GET, PUT, and DELETE methods.

Network

A network is an isolated virtual Layer 2 broadcast domain that is typically reserved for the tenant who created it, unless the tenant configures the network to be shared. Tenants can create multiple networks until they reach the thresholds per-tenant quota.

In the Cloud Networks 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 3.1. 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:

Attribute	Type	Required	CRUD[a]	Default Value	Validation Constraints	Notes
						• 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	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.

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. IP addresses can be selected either from the whole subnet CIDR or from allocation pools that are 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.

The following table describes the attributes for subnet objects.

Table 3.2. 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	cidr representing IP range for this subnet, based on IP version

Attribute	Type	Required	CRUD[a]	Default Value	Validation Constraints	Notes
					<network_address>/<prefix>	
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	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.

Port

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

The following table describes the attributes for port objects.

Table 3.3. 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.

Attribute	Type	Required	CRUD[a]	Default Value	Validation Constraints	Notes
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	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.

Router

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 associated subnet. By default, the IP address of an 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. See the section called “Extension - Layer-3 Networking (Router)” [88] for the associated API operations.

The following table describes the attributes for router objects.

Table 3.4. Router Attributes

Attribute	Type	Description
id	uuid-str	The unique identifier for the router.
name	String	The human-readable name for the router. The name might not be unique.
admin_state_up	Bool	The administrative state of router (`True` - router is up or `False` - router is down).
status	String	Indicates whether a router is currently operational or not..
tenant_id	uuid-str	The owner of router. Only admin users can specify a tenant_id other than their own.
external_gateway_info	dict	Information about the external gateway for the router.

Glossary

HTTP
: HyperText Transfer Protocol. The protocol that tells browsers where to find information.

PublicNet
: Provides access to the Internet, Rackspace services such as Cloud Monitoring, Managed Cloud Service Level, RackConnect, Cloud Backup, and certain operating system updates. When you list networks through Cloud Networks, PublicNet is labeled `public`.

REST
: REpresentational State Transfer. A style of architecture for hypermedia systems that is used for the World Wide web.

RESTful
: A kind of web service API that uses REST.

server
: A computer that provides explicit services to the client software running on that system. A server is a virtual machine (VM) instance in the Cloud Servers environment. To create a server, you must specify a name, flavor reference, and image reference.

ServiceNet
: Provides access to Rackspace services such as Cloud Files, Cloud Databases, and Cloud Backup, and to certain packages and patches through an internal only, multi-tenant network connection within each Rackspace data center. When you list networks through Cloud Networks, ServiceNet is labeled as `private`.

4. General API Information

The Cloud Networks API v2.0 is a RESTful HTTP service that uses all aspects of the HTTP protocol including methods, URIs, media types, and response codes. This section provides more information about authentication, cURL, and filtering and pagination of requests.

Get Authentication Token

To authenticate access to Rackspace Cloud services, you issue an authentication request to the Rackspace Cloud Identity Service, which is an implementation of the OpenStack Keystone Identity Service v2.0.

Important

Multiple Rackspace Cloud Identity Service endpoints exist. You may use any endpoint, regardless of where your account was created.

Use the chosen endpoint, as follows:

National location	Rackspace Cloud Identity Service endpoint
US	https://identity.api.rackspacecloud.com/v2.0
UK	https://lon.identity.api.rackspacecloud.com/v2.0

In response to valid credentials, an authentication request to the Rackspace Cloud Identity Service returns an authentication token and a service catalog that contains a list of all services and endpoints available for this token. Because the authentication token expires after 24 hours, you must generate a token once a day.

For detailed information about the OpenStack Keystone Identity Service v2.0, see *Cloud Identity Client Developer Guide API v2.0*. For information about support for legacy identity endpoints, see Alternate Authentication Endpoints.

Authenticate with cURL

To authenticate, you issue a **POST /tokens** request to the appropriate Rackspace Cloud Identity Service endpoint.

In the request body, supply one of the following sets of credentials:

- user name and password
- user name and API key

Your username and password are the ones that you use to log into the Rackspace Cloud Control Panel.

To obtain your API key, log into http://mycloud.rackspace.com, click your username, and select **API Keys** to get your key.

To authenticate with cURL:

1. Get an authentication token and a service catalog with a list of endpoints.

   Use one of the following cURL commands to authenticate to the US Identity endpoint:

Example 4.1. Authenticate with Username and Password: JSON Request

```
$ curl -s https://identity.api.rackspacecloud.com/v2.0/tokens -X 'POST' \
       -d '{"auth":{"passwordCredentials":{"username":"theUserName", "password":"thePassword"}}}' \
       -H "Content-Type: application/json" | python -m json.tool
```

Example 4.2. Authenticate with Username and API Key: JSON Request

```
$ curl -s https://identity.api.rackspacecloud.com/v2.0/tokens -X 'POST' \
       -d '{"auth":{"RAX-KSKEY:apiKeyCredentials":{"username":"theUserName", "apiKey":"0000000000000000000"}}}' \
       -H "Content-Type: application/json" | python -m json.tool
```

Note

For information about the json.tool, see Note about JSON tool in *Cloud Networks™ Getting Started*.

In response to valid credentials, your request returns an authentication token and a service catalog with the endpoints to request services.

Do not include explicit API endpoints in your scripts and applications. Instead, find the endpoint for your service and region.

The following output shows a partial authentication response in JSON format:

Example 4.3. Authenticate: JSON Response

```
{
    "access": {
        "serviceCatalog": [
            {
                "endpoints": [
                    {
                        "publicURL": "https://ord.loadbalancers.api.rackspacecloud.com/v1.0/010101",
                        "region": "ORD",
                        "tenantId": "010101"
                    },
                    {
                        "publicURL": "https://dfw.loadbalancers.api.rackspacecloud.com/v1.0/010101",
                        "region": "DFW",
                        "tenantId": "010101"
```

```
                    }
                ],
                "name": "cloudLoadBalancers",
                "type": "rax:load-balancer"
            },
            {
                "endpoints": [
                    {
                        "publicURL": "https://monitoring.api.rackspacecloud.com/v1.0/010101",
                        "tenantId": "010101"
                    }
                ],
                "name": "cloudMonitoring",
                "type": "rax:monitor"
            },
            {
                "endpoints": [
                    {
                        "publicURL": "https://preprod.dfw.servers.api.rackspacecloud.com/v2/010101",
                        "region": "DFW",
                        "tenantId": "010101"
                    }
                ],                    ❶
                "name": "cloudServersPreprod",
                "type": "compute"
            },
            {                                          ❷
                "endpoints": [
                    {
                        "internalURL": "https://snet-storage101.dfw1.clouddrive.com/v1/
MossoCloudFS_530f8649-324c-499c-a075-2195854d52a7",
                        "publicURL": "https://storage101.dfw1.clouddrive.com/v1/MossoCloudFS_530f8649-324c-499c-
a075-2195854d52a7",
                        "region": "DFW",
                        "tenantId": "MossoCloudFS_530f8649-324c-499c-a075-2195854d52a7"
                    }
                ],
                "name": "cloudFiles",
                "type": "object-store"
            },
            {
                "endpoints": [
                    {                                  ❸
                        "publicURL": "https://servers.api.rackspacecloud.com/v1.0/010101",
                        "tenantId": "010101",
                        "versionId": "1.0",
                        "versionInfo": "https://servers.api.rackspacecloud.com/v1.0",
                        "versionList": "https://servers.api.rackspacecloud.com/"
                    }                                  ❹
                ],                                     ❺
                "name": "cloudServers",
                "type": "compute"
            },
            {
                "endpoints": [
                    {
                        "publicURL": "https://dfw.servers.api.rackspacecloud.com/v2/010101",
                        "region": "DFW",
                        "tenantId": "010101",
                        "versionId": "2",
                        "versionInfo": "https://dfw.servers.api.rackspacecloud.com/v2",
                        "versionList": "https://dfw.servers.api.rackspacecloud.com/"
                    },
                    {
                        "publicURL": "https://ord.servers.api.rackspacecloud.com/v2/010101",
                        "region": "ORD",
                        "tenantId": "010101",
                        "versionId": "2",
                        "versionInfo": "https://ord.servers.api.rackspacecloud.com/v2",
                        "versionList": "https://ord.servers.api.rackspacecloud.com/"
                    }
                ],
                "name": "cloudServersOpenStack",
                "type": "compute"
            },
            {
                "endpoints": [
                    {
                        "publicURL": "https://dns.api.rackspacecloud.com/v1.0/010101",
                        "tenantId": "010101"
                    }
                ],
                "name": "cloudDNS",
                "type": "rax:dns"
            },
            {
                "endpoints": [
                    {
                        "publicURL": "https://ord.databases.api.rackspacecloud.com/v1.0/010101",
                        "region": "ORD",
                        "tenantId": "010101"
```

```
                    },
                    {
                        "publicURL": "https://dfw.databases.api.rackspacecloud.com/v1.0/010101",
                        "region": "DFW",
                        "tenantId": "010101"
                    }
                ],
                "name": "cloudDatabases",
                "type": "rax:database"
            },
            {
                "endpoints": [
                    {
                        "publicURL": "https://cdn1.clouddrive.com/v1/MossoCloudFS_530f8649-324c-499c-
a075-2195854d52a7",
                        "region": "DFW",
                        "tenantId": "MossoCloudFS_530f8649-324c-499c-a075-2195854d52a7"
                    }
                ],
                "name": "cloudFilesCDN",
                "type": "rax:object-cdn"
            }
        ],
        "token": {
            "expires": "2012-08-14T12:31:16.000-05:00",
            "id": "459a28e0-777f-416c-8f22-9f6598fabd2f",
            "tenant": {
                "id": "010101",
                "name": "010101"
            }
        },
        "user": {
            "id": "01010156",
            "name": "MyRackspaceAcct",
            "roles": [
                {
                    "description": "User Admin Role.",
                    "id": "3",
                    "name": "identity:user-admin"
                }
            ]
        }
    }
}
```

Successful authentication returns the following information:

❶ **Endpoints to request Rackspace Cloud services**. Appears in the `endpoints` element in the `serviceCatalog` element.

Endpoints information includes the public URL, which is the endpoint that you use to access the service, region, tenant ID, and version information.

To access the Cloud Networks or next generation Cloud Servers service, use the endpoint for the `cloudServersOpenStack` service.

❷ **Tenant ID**. Appears in the `tenantId` field in the `endpoints` element. Also known as the account number.

You include the tenant ID in the endpoint URL when you call a Cloud service.

In the following example, you export the tenant ID, `010101`, to the `account` environment variable and the authentication token to the `token` environment variable. Then, you issue a cURL command, as follows:

```
$ export account="010101"
$ export token="00000000-0000-0000-000000000000"
$ curl -s https://dfw.servers.api.rackspacecloud.com/v2/$account/images/detail \
    -H "X-Auth-Token: $token" | python -m json.tool
```

❸ **The name of the service**. Appears in the `name` field.

Locate the correct service name in the service catalog, as follows:

- **First generation Cloud Servers**. Named `cloudServers` in the catalog.

If you use the authentication token to access this service, you can view and perform first generation Cloud Servers API operations against your first generation Cloud Servers.

- **Cloud Networks or next generation Cloud Servers**. Named `cloudServersOpenStack` in the catalog.

  To access the Cloud Networks or next generation Cloud Servers service, use the `publicURL` value for the `cloudServersOpenStack` service.

  Might show multiple endpoints to enable regional choice. Select the appropriate endpoint for the region that you want to interact with by examining the `region` field.

  If you use the authentication token to access this service, you can view and perform Cloud Networks or next generation Cloud Servers API operations against your next generation Cloud Servers. To complete Cloud Networks API operations, you must also get access to this service. To request access, click here.

❹ **Expiration date and time for authentication token**. Appears in the `expires` field in the `token` element.

After this date and time, the token is no longer valid.

This field predicts the maximum lifespan for a token, but does not guarantee that the token reaches that lifespan.

Clients are encouraged to cache a token until it expires.

Because the authentication token expires after 24 hours, you must generate a token once a day.

❺ **Authentication token**. Appears in the `id` field in the `token` element.

You pass the authentication token in the `X-Auth-Token` header each time that you send a request to a service.

In the following example, you export the tenant ID, `010101`, to the `account` environment variable. You also export the authentication token, `00000000-0000-0000-000000000000`, to the `token` environment variable. Then, you issue a cURL command, as follows:

```
$ export account="010101"
$ export token="00000000-0000-0000-000000000000"
$ curl -s https://dfw.servers.api.rackspacecloud.com/v2/$account/images/detail \
    -H "X-Auth-Token: $token" | python -m json.tool
```

2. Copy the values in the `publicURL` and `tenantId` fields for the `cloudServersOpenStack` service for your region.

   Copy the authentication token from the `id` field in the `token` element.

   In the next step, you set environment variables to these values.

Role Based Access Control

Role Based Access Control (RBAC) restricts access to the capabilities of Rackspace Cloud services, including the Cloud Networks API, to authorized users only. RBAC enables Rackspace Cloud customers to specify which account users of their Cloud account have access to which Cloud Networks API service capabilities, based on roles defined by Rackspace (see Table 4.2, " Multiproduct (Global) Roles and Permissions" [20]). The permissions to perform certain operations in Cloud Networks API – create, read, update, delete – are assigned to specific roles. The account owner user assigns these roles, either global (multiproduct) or product-specific (for example, Cloud Networks), to account users.

Assigning Roles to Account Users

The account owner (identity:user-admin) can create account users on the account and then assign roles to those users. The roles grant the account users specific permissions for accessing the capabilities of the Cloud Networks service. Each account has only one account owner, and that role is assigned by default to any Rackspace Cloud account when the account is created.

See the *Cloud Identity Client Developer Guide* for information about how to perform the following tasks:

- *Create account users*
- *Assign roles to account users*
- *Delete roles from account users*

Note

The account owner (identity:user-admin) role cannot hold any additional roles because it already has full access to all capabilities.

Roles Available for Cloud Networks

Three roles (observer, creator, and admin) can be used to access the Cloud Networks API specifically. The following table describes these roles and their permissions.

Table 4.1. Cloud Networks Product Roles and Permissions

Role name	Role permissions
cn:admin	This role provides Create, Read, Update, and Delete permissions in Cloud Networks, where access is granted.
cn:creator	This role provides Create, Read, and Update permissions in Cloud Networks, where access is granted.
cn:observer	This role provides Read permission in Cloud Networks, where access is granted.

Additionally, two multiproduct roles apply to all products. Users with multiproduct roles inherit access to future products when those products become RBAC-enabled. The following table describes these roles and their permissions.

Table 4.2. Multiproduct (Global) Roles and Permissions

Role Name	Role Permissions
admin	This role provides Create, Read, Update, and Delete permissions in all products, where access is granted.
observer	This role provides Read permission in all products, where access is granted.

Resolving Conflicts between RBAC Multiproduct and Custom (Product-Specific) Roles

The account owner can set roles for both multiproduct and Cloud Networks scope, and it is important to understand how any potential conflicts among these roles are resolved. When two roles appear to conflict, the role that provides the more extensive permissions takes precedence. Therefore, admin roles take precedence over observer and creator roles, because admin roles provide more permissions.

The following table shows two examples of how potential conflicts between user roles in the Control Panel are resolved:

Permission Configuration	View of Permission in the Control Panel	Can the User Perform Product Admin Functions in the Control Panel?
User is assigned the following roles: multiproduct **observer** and Cloud Networks **admin**	Appears that the user has only the multiproduct **observer** role	Yes, for Cloud Networks only. The user has the **observer** role for the rest of the products.
User is assigned the following roles: multiproduct **admin** and Cloud Networks **observer**	Appears that the user has only the multiproduct **admin** role	Yes, for all of the products. The Cloud Networks **observer** role is ignored.

RBAC Permissions Cross-Reference to Cloud Networks API Operations

API operations for Cloud Networks may or may not be available to all roles. To see which operations are permitted to invoke which calls, please see the *Permissions Matrix for Cloud Networks* article in the Rackspace Knowledge Center.

How cURL Commands Work

cURL is a command-line tool that you can use to interact with *REST* interfaces. cURL lets you to transmit and receive *HTTP* requests and responses from the command line or a shell script, which enables you to work with the API directly. It is available for Linux distributions, Mac OS X, and Windows. For information about cURL, see http://curl.haxx.se/.

To use XML requests and responses, see the section called "XML Requests and Responses" [24].

To run the cURL request examples shown in this guide, copy each example from the HTML version of this guide directly to the command line or a script.

The following command is an example cURL command that provisions a server with an isolated network:

Example 4.4. cURL Command Example: JSON Request

```
          $ curl https://dfw.servers.api.rackspacecloud.com/v2/$account/servers \
      -X POST \
      -H "X-Auth-Project-Id: $account" \
      -H "Content-Type: application/json" \
      -H "Accept: application/json" \
      -H "X-Auth-Token: $token" \
      -d '{"server": {"name": "my_server_with_network", "imageRef": "d42f821e-c2d1-4796-9f07-
af5ed7912d0e", "flavorRef": "2", "max_count": 1, "min_count": 1, "networks": [{"uuid":
 "538a112a-34d1-47ff-bf1e-c40639e886e2"}, {"uuid": "00000000-0000-0000-0000-000000000000"}, {"uuid":
 "11111111-1111-1111-1111-111111111111"}]}}' \
     | python -m json.tool
```

Note

The carriage returns in the cURL request examples use a backslash (\) as an escape character. The escape character allows continuation of the command across multiple lines. However, do not include the escape character in the JSON or XML request body within the cURL command.

The cURL examples in this guide use the following command-line options:

Table 4.3. cURL Command-Line Options

Option	Description
`-d`	Sends the specified data in a **POST** request to the HTTP server. Use this option to send a JSON or XML request body to the server.

`-H`	Specifies an extra HTTP header in the request. You can specify any number of extra headers. Precede each header with the `-H` option.
	
	Common headers in Rackspace API requests are as follows:
	
	• `Content-Type`. Required for operations with a request body.
	
	Specifies the format of the request body. Following is the syntax for the header where *`format`* is either `json` or `xml`.
	
	`Content-Type: application/format`
	
	• `X-Auth-Token`. Required. Specifies the authentication token.
	
	• `X-Auth-Project-Id`. Optional. Specifies the project ID, which can be your account number or another value.
	
	• `Accept`. Optional.
	
	Specifies the format of the response body. Following is the syntax for the header where *`format`* is either `json` or `xml`. The default is `json`.
	
	`Accept: application/format`

`-i`	Includes the HTTP header in the output.
`-s`	Specifies silent or quiet mode, which makes cURL mute. No progress or error messages are shown.
`-T`	Transfers the specified local file to the remote URL.
`-X`	Specifies the request method to use when communicating with the HTTP server. The specified request is used instead of the default method, which is **GET**.

Note

For commands that return a response, you can append the following code to the command to call json.tool to pretty-print output:

```
| python -m json.tool
```

To use json.tool, import the json module. For information about json.tool, see json — JSON encoder and decoder.

If you run a Python version older than 2.6, import the simplejson module and use simplejson.tool. For information about simplejson.tool, see simplejson — JSON encoder and decoder.

If you do not want to pretty-print JSON output, omit this code.

XML Requests and Responses

The following example shows a cURL command that specifies an XML request body and returns an XML response. The command creates a server.

Example 4.5. cURL Command Example: XML Request

```
$ curl -i https://dfw.servers.api.rackspacecloud.com/v2/$account/servers.xml❶ \
     -X POST \
     -H "X-Auth-Project-Id: $account" \
     -H "Content-Type: application/xml" \  ❷
     -H "Accept: application/xml" \  ❸
     -H "X-Auth-Token: $token" \
     -T server_post_req.xml❹ | ppxml ❺
```

This example includes the following changes to the basic JSON request format in Example 4.4, “cURL Command Example: JSON Request” [21]:

❶ The endpoint in the cURL command has `.xml` appended to it to return an XML response.

❷ The `Content-Type:` header specifies `application/xml` instead of `application/json`.

❸ The `Accept:` header specifies `application/xml` instead of `application/json`.

❹ The request body, if required, should be specified it in XML format. In this example, the XML body is passed in the `server_post_req.xml` file, as follows:

```
                  <?xml version="1.0" encoding="UTF-8"?>
<server xmlns="http://docs.openstack.org/compute/api/v1.1" imageRef="3afe97b2-26dc-49c5-a2cc-a2fc8d80c001"
 flavorRef="2"
  name="api-test-server-xml2">
  <metadata>
    <meta key="My Server Name">API Test Server XML</meta>
  </metadata>
  <personality>
    <file path="/etc/banner.txt"> ICAgICAgDQoiQSBjbG91ZCBkb2VzIG5vdCBrbm93IHdoeSBp
      dCBtb3ZlcyBpbiBqdXN0IHN1Y2ggYSBkaXJlY3Rpb24gYW5k IGF0IHN1Y2ggYSBzcGVlZC4uLkl0IGZlZWxzIGFuIGltcHVs
      c2lvbi4uLnRoaXMgaXMgdGhlIHBsYWNlIHRvIGdvIG5vdy4g QnV0IHRoZSBza3kga25vd3MgdGhlIHJlYXNvbnMgYW5kIHRo
      ZSBwYXR0ZXJucyBiZWhpbmQgYWxsIGNsb3VkcywgYW5kIHlv dSB3aWxsIGtub3csIHRvbywgd2hlbiB5b3UgbGlmdCB5b3Vy
      c2VsZiBoaWdoIGVub3VnaCB0byBzZWUgYmV5b25kIGhvcml6 b25zLiINCg0KLVJpY2hhcmQgQmFjaA==</file>
  </personality>
  <networks>
    <network uuid="0ef47ac7-6797-4e01-8a47-ed26ec3aaa56"/>
    <network uuid="00000000-0000-0000-0000-000000000000"/>
    <network uuid="11111111-1111-1111-1111-111111111111"/>
  </networks>
  <keypair>
    <key_name>name_of_keypair-96bbe50e-05e1-4d59-9115-4779a3ebcc2e</key_name>
  </keypair>
</server>
```

❺ To pretty-print the XML output, set the `ppxml` alias, as follows:

```
                  $ alias ppxml='python -c "import sys, xml.dom.minidom; print xml.dom.minidom.parseString(sys.stdin.
read()).toprettyxml()"'
```

Then, append the `ppxml` alias to the cURL command.

The cURL command returns information about the new server in XML format, as shown in the following example.

Example 4.6. cURL Command Example: XML Response

```
           <?xml version='1.0' encoding='UTF-8'?>
<server
    xmlns:OS-DCF="http://docs.openstack.org/compute/ext/disk_config/api/v1.1"
    xmlns:atom="http://www.w3.org/2005/Atom"
    xmlns="http://docs.openstack.org/compute/api/v1.1"
    id="ed5c7754-29b6-45fa-96cb-ab64958c8c0a" adminPass="Dd5pNZtpVVQ3"
```

```
    OS-DCF:diskConfig="AUTO">
    <metadata/>
    <atom:link
        href="https://dfw.servers.api.rackspacecloud.com/v2/010101/servers/ed5c7754-29b6-45fa-96cb-
ab64958c8c0a"
        rel="self"/>
    <atom:link
        href="https://dfw.servers.api.rackspacecloud.com/010101/servers/ed5c7754-29b6-45fa-96cb-
ab64958c8c0a"
        rel="bookmark"/>
</server>
```

Filtering Requests

The Cloud Networks API 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 API returns only objects that meet all filtering criteria. The operation applies an **AND** condition among the filters. There is no **OR** condition.

By default, the API returns all attributes for any show or list call. You can use the *fields* query parameter to control the attributes that the API returns.

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

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

Pagination of Requests

To reduce load on the service, list operations returns a maximum number of items at a time. To help you navigate the collection, you can set the *limit*, *marker*, and *page_reverse* parameters in the URI.

For example, the following request returns up to 100 pages, starting with the item with ID=1234:

```
?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* does not contain a `next` link, and the last page in the list requested with *page_reverse=True* does not contain a `previous` link. The following examples illustrate two pages with three items.

Example 4.7. Network Collection, First Page: JSON or XML Request

```
GET /v2.0/networks.json?limit=2 HTTP/1.1
```

Example 4.8. 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&amp;marker=71c1e68c-171a-4aa2-
aca5-50ea153a3718",
            "rel": "next"
        },
        {
            "href": "http://127.0.0.1:9696/v2.0/networks.json?limit=2&amp;marker=
396f12f8-521e-4b91-8e21-2e003500433a&amp;page_reverse=True",
            "rel": "previous"
        }
    ]
}
```

Example 4.9. 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>
```

Example 4.10. Network Collection, Last Page: JSON or XML Request

```
GET /v2.0/networks.json?limit=2&marker=71c1e68c-171a-4aa2-aca5-50ea153a3718
 HTTP/1.1
```

Example 4.11. 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&amp;marker=
b3680498-03da-4691-896f-ef9ee1d856a7&amp;page_reverse=True",
                "rel": "previous"
            }
        ]
}
```

Example 4.12. 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;amp;marker=b3680498-03da-4691-896f-
ef9ee1d856a7&amp;amp;page_reverse=True"
        rel="previous"/>
</networks>
```

5. API Operations - Neutron

The Rackspace Cloud Networks API v2 serves the following functions:

- Provides virtual networking services among devices that are managed by the Rackspace Cloud Servers service.
- 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. You can choose a specific IP address from the block or let the Cloud Networks service choose the first available IP address.

Networks

This section describes the API operations for 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.
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.

This operation lists networks to which the specified tenant has access. You can control which attributes are returned by using the *`fields`* query parameter, and you can specify how many results to return per page. For information, see Filtering Requests or Pagination of Requests in the *Cloud Networks API Developer Guide*.

Normal response codes: 200

Error response codes: unauthorized (401)

Request

This operation does not require a request body.

Response

This list shows the body parameters for the response:

- **parameters**:
  - **networks**: Array. Required.

    The array of networks.
    - **status**: Xsd:string. Required.

      The network status.
    - **subnets**: Xsd:dict. Required.

      The associated subnets.
    - **name**: Xsd:string. Required.

      The network name.
    - **admin_state_up**: Xsd:bool. Required.

      The administrative state of the network (`True` if up or `False` if down).
    - **tenant_id**: Csapi:uuid. Required.

      The tenant ID.
    - **id**: Csapi:uuid. Required.

      The network ID.

- **shared**: Xsd:bool. Required.

  Indicates whether this network is shared across all tenants.

Example 5.1. 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 5.2. 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 creates a network. The tenant ID that you specify in the URI is the tenant who 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

This list shows the body parameters for the request:

- **parameters**:
  - **admin_state_up**: Xsd:bool. Optional.

    The administrative state of the network (`True` if up or `False` if down).
  - **name**: Xsd:string. Required.

    The network name.
  - **shared**: Xsd:bool. Optional.

    Admin only. Indicates whether this network is shared across all tenants.
  - **tenant_id**: Csapi:uuid. Optional.

    The ID of the tenant who owns the network. Only admin users can specify a tenant ID other than their own. You cannot change this value through authorization policies.

Example 5.3. Create Network: JSON request

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

Example 5.4. Create Network: XML request

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

Response

This list shows the body parameters for the response:

- **parameters**:
  - **networks**: Array. Required.

    The array of networks.
    - **status**: Xsd:string. Required.

      The network status.
    - **subnets**: Xsd:dict. Required.

      The associated subnets.
    - **name**: Xsd:string. Required.

      The network name.
    - **admin_state_up**: Xsd:bool. Required.

      The administrative state of the network (`True` if up or `False` if down).
    - **tenant_id**: Csapi:uuid. Required.

      The tenant ID.
    - **id**: Csapi:uuid. Required.

      The network ID.
    - **shared**: Xsd:bool. Required.

      Indicates whether this network is shared across all tenants.

Example 5.5. 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 5.6. 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.

This operation creates multiple networks with one request. In the request body, specify a list of networks.

Note

Bulk create operations are always atomic, meaning that either all or no networks in the request body are created.

Normal response codes: 201

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

Request

This list shows the body parameters for the request:

- **parameters**:
  - **admin_state_up**: Xsd:bool. Optional.

    The administrative state of the network (`True` if up or `False` if down).
  - **name**: Xsd:string. Required.

    The network name.
  - **shared**: Xsd:bool. Optional.

    Admin only. Indicates whether this network is shared across all tenants.
  - **tenant_id**: Csapi:uuid. Optional.

    The ID of the tenant who owns the network. Only admin users can specify a tenant ID other than their own. You cannot change this value through authorization policies.

Example 5.7. Bulk Create Networks: JSON request

```
{
   "networks":[
      {
         "name":"sample_network_1",
         "admin_state_up":false
      },
      {
         "name":"sample_network_2",
         "admin_state_up":false
      }
   ]
```

```
}
```

Example 5.8. Bulk Create Networks: XML request

```
<?xml version="1.0" encoding="UTF-8"?>
<networks>
    <network>
        <name>sample_network_1</name>
    </network>
    <network>
        <name>sample_network_2</name>
    </network>
</networks>
```

Response

This list shows the body parameters for the response:

- **parameters**:
  - **networks**: Array. Required.

    The array of networks.
    - **status**: Xsd:string. Required.

      The network status.
    - **subnets**: Xsd:dict. Required.

      The associated subnets.
    - **name**: Xsd:string. Required.

      The network name.
    - **admin_state_up**: Xsd:bool. Required.

      The administrative state of the network (`True` if up or `False` if down).
    - **tenant_id**: Csapi:uuid. Required.

      The tenant ID.
    - **id**: Csapi:uuid. Required.

      The network ID.
    - **shared**: Xsd:bool. Required.

      Indicates whether this network is shared across all tenants.

Example 5.9. 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 5.10. 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>60cd4f6dbc2f491982a284e7b83b5be3</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.

This operation displays details for a specified network. You can control which attributes are returned by using the *`fields`* query parameter. For information, see Filtering Requests in the *Cloud Networks API Developer Guide*.

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.

This operation does not require a request body.

Response

This list shows the body parameters for the response:

- **parameters**:

  - **networks**: Array. Required.

    The array of networks.

    - **status**: Xsd:string. Required.

      The network status.

    - **subnets**: Xsd:dict. Required.

      The associated subnets.

    - **name**: Xsd:string. Required.

      The network name.

    - **admin_state_up**: Xsd:bool. Required.

      The administrative state of the network (`True` if up or `False` if down).

    - **tenant_id**: Csapi:uuid. Required.

      The tenant ID.

- **id**: Csapi:uuid. Required.

  The network ID.

- **shared**: Xsd:bool. Required.

  Indicates whether this network is shared across all tenants.

Example 5.11. 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 5.12. 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.

This operation updates an existing 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.

This list shows the body parameters for the request:

- **parameters**:
  - **admin_state_up**: Xsd:bool. Optional.

    The administrative state of the network (`True` if up or `False` if down).
  - **name**: Xsd:string. Required.

    The network name.
  - **shared**: Xsd:bool. Optional.

    Admin only. Indicates whether this network is shared across all tenants.
  - **tenant_id**: Csapi:uuid. Optional.

    The ID of the tenant who owns the network. Only admin users can specify a tenant ID other than their own. You cannot change this value through authorization policies.

Example 5.13. Update Network: JSON request

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

Example 5.14. 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

This list shows the body parameters for the response:

- **parameters**:
  - **networks**: Array. Required.

    The array of networks.
    - **status**: Xsd:string. Required.

      The network status.
    - **subnets**: Xsd:dict. Required.

      The associated subnets.
    - **name**: Xsd:string. Required.

      The network name.
    - **admin_state_up**: Xsd:bool. Required.

      The administrative state of the network (`True` if up or `False` if down).
    - **tenant_id**: Csapi:uuid. Required.

      The tenant ID.
    - **id**: Csapi:uuid. Required.

      The network ID.
    - **shared**: Xsd:bool. Required.

      Indicates whether this network is shared across all tenants.

Example 5.15. 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 5.16. 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.

This operation deletes an existing 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.

This operation does not require a request body.

Subnets

This section describes the API operations for subnets.

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

List Subnets

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

This operation list subnets to which the specified tenant has access. Default policy settings return only subnets owned by the tenant submitting the request, unless the request is submitted by a user with administrative rights. You can control which attributes are returned by using the *fields* query parameter, and you can specify how many results to return per page. For information, see Filtering Requests or Pagination of Requests in the *Cloud Networks API Developer Guide*.

Normal response codes: 200

Error response codes: unauthorized (401)

Request

This table shows the URI parameters for the list subnets request:

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

This operation does not require a request body.

Response

This list shows the body parameters for the response:

- **parameters**:

  - **subnets**: Array. Required.

    The array of subnets.

    - **allocation_pools**: Array. Required.

      The array of subnet allocation pools.

      - **end**: Xsd:string. Required.

        The ending IP address of the subnet allocation pool.

      - **start**: Xsd:string. Required.

        The starting IP address of the subnet allocation pool.

    - **cidr**: Xsd:string. Required.

      The CIDR for the subnet.

- **dns_nameservers**: Array. Required.

  The array of dns name servers for the subnet.

- **enable_dhcp**: Xsd:boolean. Required.

  Indicates if DHCP for the subnet is enabled (`True`) or disabled (`False`).

- **gateway_ip**: Xsd:string. Required.

  The gateway IP address for the subnet.

- **host_routes**: Array. Required.

  The array of host routes for the subnet.

- **id**: Csapi:uuid. Required.

  The ID of the subnet.

- **ip_version**: Xsd:string. Required.

  The subnet IP version, which is 4 or 6.

- **name**: Xsd:string. Required.

  The subnet name.

- **network_id**: Csapi:uuid. Required.

  The ID of the subnet.

- **tenant_id**: Csapi:uuid. Required.

  The ID of the tenant who owns the subnet. Only admin users can specify a tenant ID other than their own.

Example 5.17. List Subnets: JSON response

```
{
   "subnets":[
      {
         "allocation_pools":[
            {
               "end":"10.0.3.254",
               "start":"10.0.3.2"
            }
         ],
         "cidr":"10.0.3.0/24",
         "dns_nameservers":[

         ],
         "enable_dhcp":true,
         "gateway_ip":"10.0.3.1",
         "host_routes":[
```

```
        ],
        "id":"91e47a57-7508-46fe-afc9-fc454e8580e1",
        "ip_version":4,
        "name":"",
        "network_id":"1a6f6006-9e0b-4f99-984c-96787ae66363",
        "tenant_id":"f667b69e4d6749749ef3bcba7251d9ce"
     },
     {
        "allocation_pools":[
           {
              "end":"10.0.2.254",
              "start":"10.0.2.2"
           }
        ],
        "cidr":"10.0.2.0/24",
        "dns_nameservers":[

        ],
        "enable_dhcp":true,
        "gateway_ip":"10.0.2.1",
        "host_routes":[

        ],
        "id":"e3c3620c-9d24-4470-b226-739da2f617c0",
        "ip_version":4,
        "name":"",
        "network_id":"1a6f6006-9e0b-4f99-984c-96787ae66363",
        "tenant_id":"f667b69e4d6749749ef3bcba7251d9ce"
     }
   ]
}
```

Example 5.18. List Subnets: XML response

```
<?xml version="1.0" encoding="UTF-8"?>
<ports>
    <port id="98017ddc-efc8-4c25-a915-774b2a633855"/>
    <port id="b832be00-6553-4f69-af33-acd554e36d08"/>
</ports>
```

This operation does not return a response body.

Create Subnet

Method	URI	Description
POST	`/v2/{tenant_id}/subnets`	Creates a subnet on a specified network.

This operation creates a subnet on a specified network. By default, Cloud Networks creates IPv4 subnets. To create an IPv6 subnet, you must specify the value 6 for the *`ip_version`* parameter in the request body. Cloud Networks does not try to derive the correct IP version from the provided CIDR. If the *`gateway_ip`* parameter is not specified, Cloud Networks 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`* parameter in the request body. If the *`allocation_pools`* parameter is not specified, Cloud Networks 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.

Warning

When you specify both the *`allocation_pools`* and *`gateway_ip`* parameters, you must 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

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

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

This list shows the body parameters for the request:

- **parameters**:
  - **subnets**: Array. Required.

    The array of subnets.
    - **cidr**: Xsd:string. Required.

      The subnet CIDR.
    - **ip_version**: Xsd:string. Required.

      The subnet IP version, which is 4 or 6.

- **network_id**: Csapi:uuid. Required.

  The network ID of the subnet.

- **name**: Xsd:string. Optional.

  The subnet name.

- **allocation_pools**: Array. Optional.

  The start and end addresses for the allocation pools.

  - **end**: Xsd:string. Optional.

    The ending IP address of the subnet allocation pool.

  - **start**: Xsd:string. Optional.

    The starting IP address of the subnet allocation pool.

- **gateway_ip**: Xsd:string. Optional.

  The subnet gateway IP address.

- **subnet**:

  - **enable_dhcp**: Xsd:boolean. Required.

    Indicates if DHCP for the subnet is enabled (`True`) or disabled (`False`).

Example 5.19. 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 5.20. 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

This list shows the body parameters for the response:

- **parameters**:
  - **subnets**: Array. Required.

    The array of subnets.
    - **allocation_pools**: Array. Required.

      The array of subnet allocation pools.
      - **end**: Xsd:string. Required.

        The ending IP address of the subnet allocation pool.
      - **start**: Xsd:string. Required.

        The starting IP address of the subnet allocation pool.
    - **cidr**: Xsd:string. Required.

      The CIDR for the subnet.
    - **dns_nameservers**: Array. Required.

      The array of dns name servers for the subnet.
    - **enable_dhcp**: Xsd:boolean. Required.

      Indicates if DHCP for the subnet is enabled (`True`) or disabled (`False`).
    - **gateway_ip**: Xsd:string. Required.

      The gateway IP address for the subnet.
    - **host_routes**: Array. Required.

      The array of host routes for the subnet.
    - **id**: Csapi:uuid. Required.

      The ID of the subnet.

- **ip_version**: Xsd:string. Required.

  The subnet IP version, which is 4 or 6.

- **name**: Xsd:string. Required.

  The subnet name.

- **network_id**: Csapi:uuid. Required.

  The ID of the subnet.

- **tenant_id**: Csapi:uuid. Required.

  The ID of the tenant who owns the subnet. Only admin users can specify a tenant ID other than their own.

Example 5.21. 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 5.22. 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.

Bulk Create Subnet

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

This operation creates multiple subnets with a single request.

Warning

Bulk create operations are always atomic, meaning that 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

This table shows the URI parameters for the bulk create subnet request:

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

This list shows the body parameters for the request:

- **parameters**:

  - **subnets**: Array. Required.

    The array of subnets.

    - **cidr**: Xsd:string. Required.

      The subnet CIDR.

    - **ip_version**: Xsd:string. Required.

      The subnet IP version, which is 4 or 6.

    - **network_id**: Csapi:uuid. Required.

      The network ID of the subnet.

    - **name**: Xsd:string. Optional.

      The subnet name.

    - **allocation_pools**: Array. Optional.

The start and end addresses for the allocation pools.

- **end**: Xsd:string. Optional.

  The ending IP address of the subnet allocation pool.

- **start**: Xsd:string. Optional.

  The starting IP address of the subnet allocation pool.

- **gateway_ip**: Xsd:string. Optional.

  The subnet gateway IP address.

- **subnet**:

  - **enable_dhcp**: Xsd:boolean. Required.

    Indicates if DHCP for the subnet is enabled (`True`) or disabled (`False`).

Example 5.23. 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 5.24. 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

This list shows the body parameters for the response:

- **parameters**:
  - **subnets**: Array. Required.

    The array of subnets.
    - **allocation_pools**: Array. Required.

      The array of subnet allocation pools.
      - **end**: Xsd:string. Required.

        The ending IP address of the subnet allocation pool.
      - **start**: Xsd:string. Required.

        The starting IP address of the subnet allocation pool.
    - **cidr**: Xsd:string. Required.

      The CIDR for the subnet.
    - **dns_nameservers**: Array. Required.

      The array of dns name servers for the subnet.
    - **enable_dhcp**: Xsd:boolean. Required.

      Indicates if DHCP for the subnet is enabled (`True`) or disabled (`False`).
    - **gateway_ip**: Xsd:string. Required.

      The gateway IP address for the subnet.
    - **host_routes**: Array. Required.

      The array of host routes for the subnet.
    - **id**: Csapi:uuid. Required.

      The ID of the subnet.
    - **ip_version**: Xsd:string. Required.

      The subnet IP version, which is 4 or 6.
    - **name**: Xsd:string. Required.

      The subnet name.
    - **network_id**: Csapi:uuid. Required.

The ID of the subnet.

- **tenant_id**: Csapi:uuid. Required.

  The ID of the tenant who owns the subnet. Only admin users can specify a tenant ID other than their own.

Example 5.25. 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 5.26. 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/{tenant_id}/subnets/{subnet_id}`	Shows information for a specified subnet.

This operation displays details about the specified subnet. You can control which attributes are returned by using the *`fields`* query parameter. For information, see Filtering Requests in the *Cloud Networks API Developer Guide*.

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
`{tenant_id}`	UUID	The tenant ID in a multi-tenancy cloud.
`{subnet_id}`	UUID	The UUID for the subnet.

This operation does not require a request body.

Response

This list shows the body parameters for the response:

- **parameters**:

  - **subnets**: Array. Required.

    The array of subnets.

    - **allocation_pools**: Array. Required.

      The array of subnet allocation pools.

      - **end**: Xsd:string. Required.

        The ending IP address of the subnet allocation pool.

      - **start**: Xsd:string. Required.

        The starting IP address of the subnet allocation pool.

    - **cidr**: Xsd:string. Required.

      The CIDR for the subnet.

    - **dns_nameservers**: Array. Required.

The array of dns name servers for the subnet.

- **enable_dhcp**: Xsd:boolean. Required.

  Indicates if DHCP for the subnet is enabled (`True`) or disabled (`False`).

- **gateway_ip**: Xsd:string. Required.

  The gateway IP address for the subnet.

- **host_routes**: Array. Required.

  The array of host routes for the subnet.

- **id**: Csapi:uuid. Required.

  The ID of the subnet.

- **ip_version**: Xsd:string. Required.

  The subnet IP version, which is 4 or 6.

- **name**: Xsd:string. Required.

  The subnet name.

- **network_id**: Csapi:uuid. Required.

  The ID of the subnet.

- **tenant_id**: Csapi:uuid. Required.

  The ID of the tenant who owns the subnet. Only admin users can specify a tenant ID other than their own.

Example 5.27. Show Subnet: JSON response

```
{
    "port":
        {
            "state": "DOWN",
            "id": "98017ddc-efc8-4c25-a915-774b2a633855"
        }
}
```

Example 5.28. Show Subnet: XML response

```
<?xml version="1.0" encoding="UTF-8"?>
<port id="98017ddc-efc8-4c25-a915-774b2a633855" state="DOWN"/>
```

This operation does not return a response body.

Update Subnet

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

This operation updates a specified subnet.

Warning

Some parameters, such as IP version (*`ip_version`*), CIDR (*`cidr`*), and IP allocation pools (*`allocation_pools`*) cannot be updated. Attempting to update these parameters 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
`{tenant_id}`	UUID	The tenant ID in a multi-tenancy cloud.
`{subnet_id}`	UUID	The UUID for the subnet.

This list shows the body parameters for the request:

- **parameters**:
  - **subnet**: Array. Required.

    The subnet details.
    - **name**: Xsd:string. Required.

      The subnet name.
    - **network_id**: Csapi:uuid. Required.

      The network ID of the subnet.
    - **tenant_id**: Csapi:uuid. Required.

      The tennant ID of the owner of the subnet.
    - **allocation_pools**: Array. Optional.

      The start and end addresses for the allocation pools.
      - **end**: Xsd:string. Optional.

The ending IP address of the subnet allocation pool.

- **start**: Xsd:string. Optional.

  The starting IP address of the subnet allocation pool.

- **gateway_ip**: Xsd:string. Optional.

  The gateway IP address.

- **ip_version**: Xsd:integer. Required.

  The IP version for the subnet, which is either 4 or 6.

- **cidr**: Xsd:string. Required.

  The CIDR.

- **id**: Csapi:uuid. Required.

  The ID of the subnet.

Example 5.29. Update Subnet: JSON request

```
status: 201,
content-length: 306,
content-type: application/json

{
  "subnet": {
    "name": "",
    "network_id": "ed2e3c10-2e43-4297-9006-2863a2d1abbc",
    "tenant_id": "c1210485b2424d48804aad5d39c61b8f",
    "allocation_pools": [{"start": "10.0.3.20", "end": "10.0.3.150"}],
    "gateway_ip": "10.0.3.1",
    "ip_version": 4,
    "cidr": "10.0.3.0/24",
    "id": "9436e561-47bf-436a-b1f1-fe23a926e031",
    "enable_dhcp": true}
}
```

Example 5.30. Update Subnet: XML request

```
<?xml version="1.0" encoding="UTF-8"?>
<port state="ACTIVE"/>
```

This operation does not require a request body.

Response

This list shows the body parameters for the response:

- **parameters**:

  - **subnets**: Array. Required.

The array of subnets.

- **allocation_pools**: Array. Required.

  The array of subnet allocation pools.

  - **end**: Xsd:string. Required.

    The ending IP address of the subnet allocation pool.

  - **start**: Xsd:string. Required.

    The starting IP address of the subnet allocation pool.

- **cidr**: Xsd:string. Required.

  The CIDR for the subnet.

- **dns_nameservers**: Array. Required.

  The array of dns name servers for the subnet.

- **enable_dhcp**: Xsd:boolean. Required.

  Indicates if DHCP for the subnet is enabled (`True`) or disabled (`False`).

- **gateway_ip**: Xsd:string. Required.

  The gateway IP address for the subnet.

- **host_routes**: Array. Required.

  The array of host routes for the subnet.

- **id**: Csapi:uuid. Required.

  The ID of the subnet.

- **ip_version**: Xsd:string. Required.

  The subnet IP version, which is 4 or 6.

- **name**: Xsd:string. Required.

  The subnet name.

- **network_id**: Csapi:uuid. Required.

  The ID of the subnet.

- **tenant_id**: Csapi:uuid. Required.

  The ID of the tenant who owns the subnet. Only admin users can specify a tenant ID other than their own.

Example 5.31. Update Subnet: JSON response

```
status: 200
content-length: 381
content-type: application/json

{
    "subnet": {
        "allocation_pools": [
            {
                "end": "10.0.3.254",
                "start": "10.0.3.2"
            }
        ],
        "cidr": "10.0.3.0/24",
        "dns_nameservers": [],
        "enable_dhcp": true,
        "gateway_ip": "10.0.3.1",
        "host_routes": [],
        "id": "91e47a57-7508-46fe-afc9-fc454e8580e1",
        "ip_version": 4,
        "name": "sample_subnet",
        "network_id": "1a6f6006-9e0b-4f99-984c-96787ae66363",
        "tenant_id": "f667b69e4d6749749ef3bcba7251d9ce"
    }
}
```

Example 5.32. Update Subnet: XML response

```
<?xml version="1.0" encoding="UTF-8"?>
<port id="98017ddc-efc8-4c25-a915-774b2a633855"/>
```

This operation does not return a response body.

Delete Subnet

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

This operation deletes a specified subnet. It fails if the 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
`{tenant_id}`	UUID	The tenant ID in a multi-tenancy cloud.
`{subnet_id}`	UUID	The UUID for the subnet.

This operation does not require a request body.

Ports

This section describes the API operations for 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.

This operation 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 a user with administrative rights. You can control which attributes are returned by using the *`fields`* query parameter, and you can specify how man results to return per page. For information, see Filtering Requests or Pagination of Requests in the *Cloud Networks API Developer Guide*.

Normal response codes: 200

Error response codes: unauthorized (401)

Request

This operation does not require a request body.

Response

This list shows the body parameters for the response:

- **parameters**:
  - **ports**: Array. Required.

    The array of ports.
    - **status**: Xsd:string. Required.

      The port status (`ACTIVE` or `DOWN`).
    - **name**: Xsd:string. Required.

      The port name.
    - **admin_state_up**: Xsd:bool. Required.

      The administrative state of the port (`True` if up or `False` if down).
    - **network_id**: Csapi:uuid. Required.

      The network ID of the port.
    - **tenant_id**: Csapi:uuid. Required.

      The ID of the tenant who owns the port. Only admin users can specify a tenant ID other than their own.
    - **binding:vif_type**: Xsd:string. Required.

The VIF type of the port.

- **device_owner**: Xsd:string. Required.

  The ID of the entity that uses this port. For example, a DHCP agent.

- **binding:capabilities**: Required.

  The binding capabilities.

  - **port_filter**: Xsd:bool. Required.

    The port filter for the binding capabilities.

- **mac_address**: Xsd:string. Required.

  The MAC address of the port.

- **fixed_ips**: Array. Required.

  Array of fixed IP addresses for the port.

  - **subnet_id**: Xsd:UUID. Required.

    The subnet ID for fixed IP address for a port.

  - **ip_address**: Xsd:string. Required.

    The IP address for fixed IP address for a port.

- **id**: Csapi:uuid. Required.

  The ID of the port.

- **device_id**: Csapi:uuid. Required.

  The ID of the device that uses this port. For example, a virtual server.

Example 5.33. List Ports: JSON response

```
{
   "ports":[
      {
         "status":"ACTIVE",
         "name":"",
         "admin_state_up":true,
         "network_id":"ebda9658-093b-41ba-80ce-0cf8cb8365d4",
         "tenant_id":"63878e4c5dd649d2a980e37aefddfa87",
         "binding:vif_type":"ovs",
         "device_owner":"compute:None",
         "binding:capabilities":{
            "port_filter":false
         },
         "mac_address":"fa:16:3e:b9:ef:05",
         "fixed_ips":[
            {
```

```
            "subnet_id":"aca4d43c-c48c-4a2c-9bb6-ba374ef7e135",
            "ip_address":"172.24.4.227"
         }
      ],
      "id":"664ebd1a-facd-4c20-948c-07a784475ab0",
      "device_id":"f288bb5f-920d-4276-8345-2c0319c16f58"
   },
   {
      "status":"DOWN",
      "name":"",
      "admin_state_up":true,
      "network_id":"ebda9658-093b-41ba-80ce-0cf8cb8365d4",
      "tenant_id":"",
      "binding:vif_type":"ovs",
      "device_owner":"network:router_gateway",
      "binding:capabilities":{
         "port_filter":false
      },
      "mac_address":"fa:16:3e:4a:3a:a2",
      "fixed_ips":[
         {
            "subnet_id":"aca4d43c-c48c-4a2c-9bb6-ba374ef7e135",
            "ip_address":"172.24.4.226"
         }
      ],
      "id":"c5ca7017-c390-4ccc-8cd7-333747e57fef",
      "device_id":"0dc517bf-9169-4aa6-88b7-569219962881"
   },
   {
      "status":"ACTIVE",
      "name":"",
      "admin_state_up":true,
      "network_id":"9d83c053-b0a4-4682-ae80-c00df269ce0a",
      "tenant_id":"625887121e364204873d362b553ab171",
      "binding:vif_type":"ovs",
      "device_owner":"network:router_interface",
      "binding:capabilities":{
         "port_filter":false
      },
      "mac_address":"fa:16:3e:2d:dc:7e",
      "fixed_ips":[
         {
            "subnet_id":"a318fcb4-9ff0-4485-b78c-9e6738c21b26",
            "ip_address":"10.0.0.1"
         }
      ],
      "id":"d7815f5b-a228-47bb-a5e5-f139c4e476f6",
      "device_id":"0dc517bf-9169-4aa6-88b7-569219962881"
   },
   {
      "status":"ACTIVE",
      "name":"",
      "admin_state_up":true,
      "network_id":"9d83c053-b0a4-4682-ae80-c00df269ce0a",
      "tenant_id":"625887121e364204873d362b553ab171",
      "binding:vif_type":"ovs",
      "device_owner":"network:dhcp",
      "binding:capabilities":{
         "port_filter":false
      },
```

```
         "mac_address":"fa:16:3e:73:6d:1c",
         "fixed_ips":[
            {
               "subnet_id":"a318fcb4-9ff0-4485-b78c-9e6738c21b26",
               "ip_address":"10.0.0.2"
            }
         ],
         "id":"f8639521-fab2-4879-94b2-83a47bee8a26",
         "device_id":"dhcpe1b8334f-9be9-5e49-aeee-b31e6df6c847-9d83c053-
b0a4-4682-ae80-c00df269ce0a"
      }
   ]
}
```

Example 5.34. 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.

This operation creates a port on a specified network. You must specify the network where the port is to be created in the *`network_id`* parameter in the request body.

Normal response codes: 201

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

Request

This list shows the body parameters for the request:

- **parameters**:

  - **port**: Required.

    The container for the port details.

    - **network_id**: Csapi:uuid. Required.

      The ID of the network for the port.

    - **admin_state_up**: Xsd:bool. Optional.

      The administrative state of the port (`True` if up or `False` if down).

    - **name**: Xsd:string. Optional.

      A symbolic name for the port.

    - **device_id**: Xsd:uuid. Optional.

      The ID of the port device.

    - **tenant_id**: Csapi:uuid. Optional.

      The ID of the tenant who owns the port. Only admin users can specify a tenant ID other than their own.

Example 5.35. Create Port: JSON request

```
{
    "port": {
        "admin_state_up": true,
        "device_id": "d6b4d3a5-c700-476f-b609-1493dd9dadc0",
        "name": "port1",
        "network_id": "6aeaf34a-c482-4bd3-9dc3-7faf36412f12"
```

```
    }
}
```

Response

This list shows the body parameters for the response:

- **parameters**:
  - **ports**: Array. Required.

    The array of ports.

    - **status**: Xsd:string. Required.

      The port status (`ACTIVE` or `DOWN`).

    - **name**: Xsd:string. Required.

      The port name.

    - **admin_state_up**: Xsd:bool. Required.

      The administrative state of the port (`True` if up or `False` if down).

    - **network_id**: Csapi:uuid. Required.

      The network ID of the port.

    - **tenant_id**: Csapi:uuid. Required.

      The ID of the tenant who owns the port. Only admin users can specify a tenant ID other than their own.

    - **binding:vif_type**: Xsd:string. Required.

      The VIF type of the port.

    - **device_owner**: Xsd:string. Required.

      The ID of the entity that uses this port. For example, a DHCP agent.

    - **binding:capabilities**: Required.

      The binding capabilities.

      - **port_filter**: Xsd:bool. Required.

        The port filter for the binding capabilities.

    - **mac_address**: Xsd:string. Required.

      The MAC address of the port.

    - **fixed_ips**: Array. Required.

Array of fixed IP addresses for the port.

- **subnet_id**: Xsd:UUID. Required.

  The subnet ID for fixed IP address for a port.

- **ip_address**: Xsd:string. Required.

  The IP address for fixed IP address for a port.

- **id**: Csapi:uuid. Required.

  The ID of the port.

- **device_id**: Csapi:uuid. Required.

  The ID of the device that uses this port. For example, a virtual server.

Example 5.36. Create Port: JSON response

```
{
   "port":[
      {
         "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":""
      }
   ]
}
```

Example 5.37. 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.

This operation creates multiple ports with a single request.

Note

Bulk create operations are always atomic, meaning that either all or no ports in the request body are created.

Normal response codes: 201

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

Request

This list shows the body parameters for the request:

- **parameters**:
  - **ports**: Array. Required.

    The array of ports.
    - **network_id**: Csapi:uuid. Required.

      The ID of the network for the port.
    - **admin_state_up**: Xsd:bool. Optional.

      The administrative state of the port (`True` if up or `False` if down).
    - **name**: Xsd:string. Optional.

      A symbolic name for the port.
    - **device_id**: Xsd:uuid. Optional.

      The ID of the port device.
    - **tenant_id**: Csapi:uuid. Optional.

      The ID of the tenant who owns the port. Only admin users can specify a tenant ID other than their own.

Example 5.38. 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 5.39. 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

This list shows the body parameters for the response:

- **parameters**:
  - **ports**: Array. Required.

    The array of ports.
    - **status**: Xsd:string. Required.

      The port status (`ACTIVE` or `DOWN`).
    - **name**: Xsd:string. Required.

      The port name.
    - **admin_state_up**: Xsd:bool. Required.

      The administrative state of the port (`True` if up or `False` if down).
    - **network_id**: Csapi:uuid. Required.

      The network ID of the port.
    - **tenant_id**: Csapi:uuid. Required.

The ID of the tenant who owns the port. Only admin users can specify a tenant ID other than their own.

- **binding:vif_type**: Xsd:string. Required.

  The VIF type of the port.

- **device_owner**: Xsd:string. Required.

  The ID of the entity that uses this port. For example, a DHCP agent.

- **binding:capabilities**: Required.

  The binding capabilities.

  - **port_filter**: Xsd:bool. Required.

    The port filter for the binding capabilities.

- **mac_address**: Xsd:string. Required.

  The MAC address of the port.

- **fixed_ips**: Array. Required.

  Array of fixed IP addresses for the port.

  - **subnet_id**: Xsd:UUID. Required.

    The subnet ID for fixed IP address for a port.

  - **ip_address**: Xsd:string. Required.

    The IP address for fixed IP address for a port.

- **id**: Csapi:uuid. Required.

  The ID of the port.

- **device_id**: Csapi:uuid. Required.

  The ID of the device that uses this port. For example, a virtual server.

Example 5.40. 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 5.41. 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.

This operation displays the details for a specified port. You can control which attributes are returned by using the *`fields`* query parameter. For information, see Filtering Requests in the *Cloud Networks API Developer Guide*.

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.

This operation does not require a request body.

Response

This list shows the body parameters for the response:

- **parameters**:
  - **ports**: Array. Required.

    The array of ports.
    - **status**: Xsd:string. Required.

      The port status (`ACTIVE` or `DOWN`).
    - **name**: Xsd:string. Required.

      The port name.
    - **admin_state_up**: Xsd:bool. Required.

      The administrative state of the port (`True` if up or `False` if down).
    - **network_id**: Csapi:uuid. Required.

      The network ID of the port.
    - **tenant_id**: Csapi:uuid. Required.

      The ID of the tenant who owns the port. Only admin users can specify a tenant ID other than their own.
    - **binding:vif_type**: Xsd:string. Required.

The VIF type of the port.

- **device_owner**: Xsd:string. Required.

  The ID of the entity that uses this port. For example, a DHCP agent.

- **binding:capabilities**: Required.

  The binding capabilities.

  - **port_filter**: Xsd:bool. Required.

    The port filter for the binding capabilities.

- **mac_address**: Xsd:string. Required.

  The MAC address of the port.

- **fixed_ips**: Array. Required.

  Array of fixed IP addresses for the port.

  - **subnet_id**: Xsd:UUID. Required.

    The subnet ID for fixed IP address for a port.

  - **ip_address**: Xsd:string. Required.

    The IP address for fixed IP address for a port.

- **id**: Csapi:uuid. Required.

  The ID of the port.

- **device_id**: Csapi:uuid. Required.

  The ID of the device that uses this port. For example, a virtual server.

Example 5.42. Show Port: JSON response

```
{
   "ports":[
      {
         "status":"ACTIVE",
         "name":"",
         "admin_state_up":true,
         "network_id":"ebda9658-093b-41ba-80ce-0cf8cb8365d4",
         "tenant_id":"63878e4c5dd649d2a980e37aefddfa87",
         "binding:vif_type":"ovs",
         "device_owner":"compute:None",
         "binding:capabilities":{
            "port_filter":false
         },
         "mac_address":"fa:16:3e:b9:ef:05",
         "fixed_ips":[
            {
```

```
            "subnet_id":"aca4d43c-c48c-4a2c-9bb6-ba374ef7e135",
            "ip_address":"172.24.4.227"
         }
      ],
      "id":"664ebd1a-facd-4c20-948c-07a784475ab0",
      "device_id":"f288bb5f-920d-4276-8345-2c0319c16f58"
   },
   {
      "status":"DOWN",
      "name":"",
      "admin_state_up":true,
      "network_id":"ebda9658-093b-41ba-80ce-0cf8cb8365d4",
      "tenant_id":"",
      "binding:vif_type":"ovs",
      "device_owner":"network:router_gateway",
      "binding:capabilities":{
         "port_filter":false
      },
      "mac_address":"fa:16:3e:4a:3a:a2",
      "fixed_ips":[
         {
            "subnet_id":"aca4d43c-c48c-4a2c-9bb6-ba374ef7e135",
            "ip_address":"172.24.4.226"
         }
      ],
      "id":"c5ca7017-c390-4ccc-8cd7-333747e57fef",
      "device_id":"0dc517bf-9169-4aa6-88b7-569219962881"
   },
   {
      "status":"ACTIVE",
      "name":"",
      "admin_state_up":true,
      "network_id":"9d83c053-b0a4-4682-ae80-c00df269ce0a",
      "tenant_id":"625887121e364204873d362b553ab171",
      "binding:vif_type":"ovs",
      "device_owner":"network:router_interface",
      "binding:capabilities":{
         "port_filter":false
      },
      "mac_address":"fa:16:3e:2d:dc:7e",
      "fixed_ips":[
         {
            "subnet_id":"a318fcb4-9ff0-4485-b78c-9e6738c21b26",
            "ip_address":"10.0.0.1"
         }
      ],
      "id":"d7815f5b-a228-47bb-a5e5-f139c4e476f6",
      "device_id":"0dc517bf-9169-4aa6-88b7-569219962881"
   },
   {
      "status":"ACTIVE",
      "name":"",
      "admin_state_up":true,
      "network_id":"9d83c053-b0a4-4682-ae80-c00df269ce0a",
      "tenant_id":"625887121e364204873d362b553ab171",
      "binding:vif_type":"ovs",
      "device_owner":"network:dhcp",
      "binding:capabilities":{
         "port_filter":false
      },
```

```
         "mac_address":"fa:16:3e:73:6d:1c",
         "fixed_ips":[
            {
               "subnet_id":"a318fcb4-9ff0-4485-b78c-9e6738c21b26",
               "ip_address":"10.0.0.2"
            }
         ],
         "id":"f8639521-fab2-4879-94b2-83a47bee8a26",
         "device_id":"dhcpe1b8334f-9be9-5e49-aeee-b31e6df6c847-9d83c053-
b0a4-4682-ae80-c00df269ce0a"
      }
   ]
}
```

Example 5.43. 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.

This operation updates a specified port. You can update information for a port, such as its symbolic name and associated IP addresses. When you update IP addresses for a port, any previously associated IP addresses are removed, returned to the appropriate subnet's allocation pools, and replaced by the IP addresses that you specified in the body of the update request. Therefore, this operation replaces the *`fixed_ip`* parameter when it is specified in the request body.

Warning

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 existing security groups are removed and replaced by the security groups specified in the body of the update request. Therefore, this operation replaces the *`security_groups`* parameter when you specify it in the request body.

Warning

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.

This list shows the body parameters for the request:

- **parameters**:
  - **port**: Required.

    The container for the port details.

    - **network_id**: Csapi:uuid. Required.

      The ID of the network for the port.

- **admin_state_up**: Xsd:bool. Optional.

  The administrative state of the port (`True` if up or `False` if down).

- **name**: Xsd:string. Optional.

  A symbolic name for the port.

- **device_id**: Xsd:uuid. Optional.

  The ID of the port device.

- **tenant_id**: Csapi:uuid. Optional.

  The ID of the tenant who owns the port. Only admin users can specify a tenant ID other than their own.

Example 5.44. Update Port: JSON request

```
{
    "port": {
        "admin_state_up": true,
        "device_id": "d6b4d3a5-c700-476f-b609-1493dd9dadc0",
        "name": "port1",
        "network_id": "6aeaf34a-c482-4bd3-9dc3-7faf36412f12"
    }
}
```

Response

This list shows the body parameters for the response:

- **parameters**:

  - **ports**: Array. Required.

    The array of ports.

    - **status**: Xsd:string. Required.

      The port status (`ACTIVE` or `DOWN`).

    - **name**: Xsd:string. Required.

      The port name.

    - **admin_state_up**: Xsd:bool. Required.

      The administrative state of the port (`True` if up or `False` if down).

    - **network_id**: Csapi:uuid. Required.

      The network ID of the port.

    - **tenant_id**: Csapi:uuid. Required.

The ID of the tenant who owns the port. Only admin users can specify a tenant ID other than their own.

- **binding:vif_type**: Xsd:string. Required.

  The VIF type of the port.

- **device_owner**: Xsd:string. Required.

  The ID of the entity that uses this port. For example, a DHCP agent.

- **binding:capabilities**: Required.

  The binding capabilities.

  - **port_filter**: Xsd:bool. Required.

    The port filter for the binding capabilities.

- **mac_address**: Xsd:string. Required.

  The MAC address of the port.

- **fixed_ips**: Array. Required.

  Array of fixed IP addresses for the port.

  - **subnet_id**: Xsd:UUID. Required.

    The subnet ID for fixed IP address for a port.

  - **ip_address**: Xsd:string. Required.

    The IP address for fixed IP address for a port.

- **id**: Csapi:uuid. Required.

  The ID of the port.

- **device_id**: Csapi:uuid. Required.

  The ID of the device that uses this port. For example, a virtual server.

Example 5.45. Update Port: JSON response


```
{
   "port":[
      {
         "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":""
      }
   ]
}
```

Example 5.46. 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.

This operation deletes the specified port. Any IP addresses that are associated with the port are returned to the appropriate subnet's 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.

This operation does not require a request body.

Extension - Layer-3 Networking (Router)

This section describes the API operations for creating and managing routers for Layer-3 (router) networking.

The Layer-3 networking extension enables Cloud Networks API users to route packets between subnets, and forward packets from internal networks to external ones.

The extension introduces the following resource:

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

See the section called "Router" [11]for more information about routers.

Method	URI	Description
GET	`/v2.0/routers`	Lists logical routers that are accessible to the tenant who submits the request.
POST	`/v2.0/routers`	Creates a logical router.
GET	`/v2.0/routers/{router_id}`	Shows details for the specified router.
PUT	`/v2.0/routers/{router_id}`	Updates the specified router.
DELETE	`/v2.0/routers/{router_id}`	Deletes the logical router and its external gateway interface.
PUT	`/v2.0/routers/{router_id}/add_router_interface`	Adds an internal interface to the specified logical router.
PUT	`/v2.0/routers/{router_id}/remove_router_interface`	Removes an internal interface from the specified logical router.

List Routers

Method	URI	Description
GET	`/v2.0/routers`	Lists logical routers that are accessible to the tenant who submits the request.

This operation lists routers for the specified tenant. Default policy settings return only those routers that are owned by the tenant who submits the request, unless an admin user submits the request. You can control which attributes are returned by using the *`fields`* query parameter, and you can specify how many results to return per page. For information, see Filtering Requests or Pagination of Requests in the *Cloud Networks API Developer Guide*.

Normal response codes: 200

Error response codes: unauthorized (401)

Request

This operation does not require a request body.

Response

This list shows the body parameters for the response:

- **status**: Xsd:string. Required.

  The router status.

- **external_gateway_information**: Xsd:dict. Required.

  The `network_id`, for the external gateway.

- **name**: Xsd:string. Required.

  The router name.

- **admin_state_up**: Xsd:bool. Required.

  The administrative state of the router. (`True` if up or `False` if down)

- **tenant_id**: Xsd:string. Required.

  The tenant ID.

- **id**: Csapi:uuid. Required.

  The router ID.

Example 5.47. 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"
        }
    ]
}
```

Create Router

Method	URI	Description
POST	`/v2.0/routers`	Creates a logical router.

This operation creates a new logical router. When it is created, a logical router does not have any internal interface, and it is not associated to any subnet. You can optionally specify an external gateway for a router at creation time. The external gateway for the router must be plugged into an external network, which has its extended field `router:external` set to `true`. To specify an external gateway, the identifier of the external network must be passed in the *`external_gateway_info`* parameter in the request body, as follows:

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

Normal response codes: 201

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

Request

This list shows the body parameters for the request:

- **name**: Xsd:string. Required.

  The router name.

- **admin_state_up**: Xsd:bool. Required.

  The administrative state of the router (`True` if up or `False` if down).

Example 5.48. Create Router: JSON request

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

Response

This list shows the body parameters for the response:

- **status**: Xsd:string. Required.

  The router status.

- **external_gateway_information**: Xsd:dict. Required.

  The `network_id`, for the external gateway.

- **name**: Xsd:string. Required.

  The router name.

- **admin_state_up**: Xsd:bool. Required.

  The administrative state of the router. (`True` if up or `False` if down)

- **tenant_id**: Xsd:string. Required.

  The tenant ID.

- **id**: Csapi:uuid. Required.

  The router ID.

Example 5.49. 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"
     }
}
```

Show Router Details

Method	URI	Description
GET	/v2.0/routers/{router_id}	Shows details for the specified router.

This operation shows details for the specified router. You can control which parameters are returned by using the *fields* query parameter, and you can specify how many results to return per page. For information, see Filtering Requests or Pagination of Requests in the *Cloud Networks API Developer Guide*.

Normal response codes: 200

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

Request

This table shows the URI parameters for the show router details request:

Name	Type	Description
{router_id}	UUID	The UUID of the router.

This operation does not require a request body.

Response

This list shows the body parameters for the response:

- **status**: Xsd:string. Required.

  The router status.

- **external_gateway_information**: Xsd:dict. Required.

  The network_id, for the external gateway.

- **name**: Xsd:string. Required.

  The router name.

- **admin_state_up**: Xsd:bool. Required.

  The administrative state of the router. (True if up or False if down)

- **tenant_id**: Xsd:string. Required.

  The tenant ID.

- **id**: Csapi:uuid. Required.

  The router ID.

Example 5.50. Show Router Details: 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"
    }
}
```

Update Router

Method	URI	Description
PUT	`/v2.0/routers/{router_id}`	Updates the specified router.

This operation updates any of the *`name`*, *`admin_state_up`*, and *`external_gateway_info`* parameters for the specified router. For more information about how to set the *`external_gateway_info`* parameter for a router, see the section called "Create Router" [91].

Note

This operation does not allow you to update the router interfaces. To update a router, use the section called "Remove Interface from Router" [101] and the section called "Add Interface to Router" [99] operations.

Normal response codes: 200

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

Request

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

Name	Type	Description
`{router_id}`	UUID	The UUID of the router.

This list shows the body parameters for the request:

- **status**: Xsd:string. Required.

  The router status.

- **external_gateway_information**: Xsd:dict. Required.

  The `network_id`, for the external gateway.

- **name**: Xsd:string. Required.

  The router name.

- **admin_state_up**: Xsd:bool. Required.

  The administrative state of the router. (`True` if up or `False` if down)

- **tenant_id**: Xsd:string. Required.

  The tenant ID.

- **id**: Csapi:uuid. Required.

  The router ID.

Example 5.51. Update Router: JSON request

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

Response

This list shows the body parameters for the response:

- **status**: Xsd:string. Required.

  The router status.

- **external_gateway_information**: Xsd:dict. Required.

  The `network_id`, for the external gateway.

- **name**: Xsd:string. Required.

  The router name.

- **admin_state_up**: Xsd:bool. Required.

  The administrative state of the router. (`True` if up or `False` if down)

- **tenant_id**: Xsd:string. Required.

  The tenant ID.

- **id**: Csapi:uuid. Required.

  The router ID.

Example 5.52. 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

Method	URI	Description
DELETE	`/v2.0/routers/{router_id}`	Deletes the logical router and its external gateway interface.

This operation deletes the specified router and, if present, its external gateway interface.

Warning

The operation fails if the router has attached interfaces.

Use the the section called "Remove Interface from Router" [101] operation to remove all router interfaces before you delete the router.

Normal response codes: 204

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

Request

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

Name	Type	Description
`{router_id}`	UUID	The UUID of the router.

This operation does not require a request body.

Add Interface to Router

Method	URI	Description
PUT	/v2.0/routers/{router_id}/add_router_interface	Adds an internal interface to the specified logical router.

This operation attaches a subnet to an internal router interface. You must specify either a *subnet_id* or a *port_id* parameter in the request body. If you specify both IDs, the operation returns a 400 Bad Request error.

If you specify a *subnet_id* parameter in the request body, the gateway IP address for the subnet is used to create the router interface.

If you specify a *port_id* parameter in the request body, the IP address associated with the port is used to create the router interface.

Warning

If either multiple IP addresses are associated with the specified port, or if no IP address is associated with the specified port, the operation returns a 400 Bad Request error.

If the port is already used, the operation returns a 409 Conflict error.

The port *id* that is returned by this operation can either be the same ID passed in the request body or the ID of a new port created by this operation to attach the specified subnet to the router. After you run this operation, the *device_id* parameter of this port is set to the ID of the router, and the *device_owner*parameter is set to network:router_interface, as shown in this 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"
   }
```

Normal response codes: 200

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

Request

This table shows the URI parameters for the add interface to router request:

Name	Type	Description
{router_id}	UUID	The UUID of the router.

This list shows the body parameters for the request:

- **subnet_id**: Csapi:uuid. Optional.

  The subnet ID.

- **port_id**: Csapi:uuid. Optional.

  The port ID.

Example 5.53. Add Interface to Router: JSON request

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

Response

This list shows the body parameters for the response:

- **subnet_id**: Csapi:uuid. Required.

  The subnet ID.

- **port_id**: Csapi:uuid. Required.

  The port ID.

Example 5.54. Add Interface to Router: JSON response

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

Remove Interface from Router

Method	URI	Description
PUT	`/v2.0/routers/{router_id}/remove_router_interface`	Removes an internal interface from the specified logical router.

This operation removes an internal router interface, which detaches a subnet from the router. It also removes the port connecting the router with the subnet from that subnet.

You must specify a *`subnet_id`* parameter, a *`port_id`*, or both *`subnet_id`* and *`port_id`* parameters in the request body, which will identify the router interface to remove.

If you specify both IDs, the *`subnet_id`* parameter must correspond to the *`subnet_id`* of the first IP address on the port specified by the *`port_id`* parameter. Otherwise, the operation returns a 409 Conflict error. The response contains information about the affected router and interface.

If the router, subnet, or port either do not exist or are not visible to you, the operation returns a 404 Not Found.

Normal response codes: 200

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

Request

This table shows the URI parameters for the remove interface from router request:

Name	Type	Description
`{router_id}`	UUID	The UUID of the router.

This list shows the body parameters for the request:

- **subnet_id**: Csapi:uuid. Optional.

  The subnet ID.

- **port_id**: Csapi:uuid. Optional.

  The port ID.

Example 5.55. Remove Interface from Router: JSON request

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

Response

This list shows the body parameters for the response:

- **id**: Csapi:uuid. Required.

  The router ID.

- **tenant_id**: Xsd:string. Required.

  The tenant ID.

- **port_id**: Csapi:uuid. Required.

  The port ID.

- **subnet_id**: Csapi:uuid. Required.

  The subnet ID.

Example 5.56. Remove Interface from Router: JSON 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"
}
```

6. API Operations - Nova-network

So what are Nova operations and why use them?

Networks

This section describes the API operations for networks.

Method	URI	Description
GET	`/os-networksv2`	Lists the networks configured for the specified tenant ID.
POST	`/os-networksv2`	Creates a network for the specified tenant ID.
POST	`/servers`	Provisions a new server with specified networks.
GET	`/os-networksv2/{id}`	Shows information for the specified network.
DELETE	`/os-networksv2/{id}`	Deletes the specified network.

List Networks

Method	URI	Description
GET	/os-networksv2	Lists the networks configured for the specified tenant ID.

This operation lists the networks configured for the tenant ID specified in the request URI.

Note

To list the networks that are associated with servers, see List Servers in the *Next Generation Cloud Servers Developer Guide*.

Normal response codes: 200

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

Request

This operation does not require a request body.

Response

This list shows the body parameters for the response:

- **parameters**:
  - **id**: Xsd:string. Required.

    The network ID.
  - **label**: Xsd:string. Required.

    The name of the network. ServiceNet is labeled as private and PublicNet is labeled as public in the network list.
  - **cidr**: Xsd:string. Required.

    The CIDR for an isolated network.

Example 6.1. List Networks: JSON response

```
{
   "networks":[
      {
         "cidr":"192.168.0.0/24",
         "id":"1f84c238-b05a-4374-a0cb-aa6140032cd1",
         "label":"new_network"
      },
      {
         "id":"00000000-0000-0000-0000-000000000000",
         "label":"public"
      },
```

```
      {
         "id":"11111111-1111-1111-1111-111111111111",
         "label":"private"
      }
   ]
}
```

Example 6.2. List Networks: XML response

```
<?xml version='1.0' encoding='UTF-8'?>
<networks>
    <network>
        <cidr>172.16.0.0/24</cidr>
        <id>e65accc0-1d98-45eb-af76-ab3d31edc7d2</id>
        <label>new_network</label>
    </network>
    <network>
        <id>00000000-0000-0000-0000-000000000000</id>
        <label>public</label>
    </network>
    <network>
        <id>11111111-1111-1111-1111-111111111111</id>
        <label>private</label>
    </network>
</networks>
```

This operation does not return a response body.

Create Networks

Method	URI	Description
POST	/os-networksv2	Creates a network for the specified tenant ID.

This operation creates a network for the tenant ID specified in the request URI.

Normal response codes: 200

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

Request

This list shows the body parameters for the request:

- **parameters**:

  - **cidr**: Capi:UUID. Required.

    The IP block from which to allocate the network. For example, **172.16.0.0/24** or **2001:DB8::/64**. For more information about CIDR notation, see *Using CIDR Notation in CLoud Networks* in the KNowledge Center.

  - **label**: Capi:UUID. Required.

    The name of the new network. For example, **my_new_network**.

Example 6.3. Create Networks: JSON request

```
{
    "network":
        {
            "cidr": "192.168.0.0/24",
            "label": "superprivate"
        }
}
```

Example 6.4. Create Networks: XML request

```
<?xml version="1.0" encoding="UTF-8"?>
<network
  cidr="192.168.0.0/24"
  label="superprivate_xml"
/>
```

This operation does not require a request body.

Response

This list shows the body parameters for the response:

- **parameters**:

- **id**: Xsd:string.

  The network ID.

- **label**: Xsd:string.

  The name of the new network.

- **cidr**: Xsd:string.

  The IP block from which the network was allocated.

Example 6.5. Create Networks: JSON response

```
{
    "network": {
        "cidr": "192.168.0.0/24",
        "id": "1ff4489e-db0e-45a6-8c9f-4616c6ef5db1",
        "label": "superprivate"
    }
}
```

Example 6.6. Create Networks: XML response

```
<network>
    <cidr>192.168.0.0/24</cidr>
    <id>f212726e-6321-4210-9bae-a13f5a33f83f</id>
    <label>superprivate_xml</label>
</network>
```

This operation does not return a response body.

Provision Servers and Attach Networks

Method	URI	Description
POST	`/servers`	Provisions a new server with specified networks.

This operation asynchronously provisions a new server. For complete information about this API operation, see Create Server in the *Next Generation Cloud Servers Developer Guide*.

You must specify the networks that you want to attach to your server. If you do not specify any networks, ServiceNet and PublicNet are attached by default. For information about the Rackspace networks, see Isolated Networks and Rackspace Networks.

You can optionally provision the server instance with specified isolated networks. However, if you specify an isolated network, you must explicitly specify the UUIDs for PublicNet and ServiceNet to attach these networks to your server. The UUID for ServiceNet is `11111111-1111-1111-1111-111111111111`, and the UUID for PublicNet is `00000000-0000-0000-0000-000000000000`. Omit these UUIDs from the request to detach from these networks.

Warning

RackConnect and Managed Cloud Service Level customers receive an error if they opt out of attaching to PublicNet or ServiceNet.

To attach a network to an existing server, you must create a virtual interface. See the section called "Create Virtual Interface" [120].

Note

To list the networks that are associated with servers, see List Servers in the *Next Generation Cloud Servers Developer Guide*.

The following table shows the status transition as the operation executes:

Status Transition:	`BUILD` → `ACTIVE`
	`BUILD` → `ERROR` (on error)

Normal response codes: 202

Error response codes: badRequest (400), unauthorized (401), forbidden (403), itemNotFound (404), badMethod (405), overLimit (413), badMediaType (415), computeFault (503), serviceUnavailable (503)

Request

This list shows the body parameters for the request:

- **parameters**:
  - **name**: String. Required.

    The server name.

- **imageRef**: String. Required.

  The image reference (either an ID or a full URL).

- **flavorRef**: String. Required.

  The flavor reference (either an ID or a full URL).

- **config_drive**: Bool. Optional.

  Is config_drive enabled (either `true` or `false`).

- **key_name**: String. Optional.

  The name of the key pair.

- **networks**: Array. Required.

  Contains network uuids.

  - **uuid**: UUID. Optional.

    By default, the server instance is provisioned with all Rackspace networks and isolated networks for the tenant. Optionally, to provision the server instance with a specific isolated tenant network, specify the UUID of the network in the **UUID** attribute. You can specify multiple UUIDs.

- **OS-DCF:diskConfig**: String. Optional.

  The disk configuration value, which is `AUTO` or `MANUAL`. See Disk Configuration Extension.

- **metadata**: Meta. Optional.

  Metadata key and value pairs.

- **personality**: Optional.

  The container for personality file path and contents.

  - **path**: String. Optional.

    The path for the personality file.

  - **contents**: File. Optional.

    The contents of the personality file.

Example 6.7. Provision Servers and Attach Networks: JSON request

```
{
    "server" : {
        "name" : "api-test-server-1",
        "imageRef" : "3afe97b2-26dc-49c5-a2cc-a2fc8d80c001",
        "flavorRef" : "2",
```

```
        "config_drive": true,
        "key_name":"name_of_keypair",
        "OS-DCF:diskConfig" : "AUTO",
        "metadata" : {
            "My Server Name" : "API Test Server 1"
        },
        "personality" : [
            {
                "path" : "/etc/banner.txt",
                "contents" : "ICAgICAgDQoiQSBjbG91ZCBkb2VzIG5vdCBrbm93IHdoeSBp
 dCBtb3ZlcyBpbiBqdXN0IHN1Y2ggYSBkaXJlY3Rpb24gYW5k
 IGF0IHN1Y2ggYSBzcGVlZC4uLkl0IGZlZWxzIGFuIGltcHVs
 c2lvbi4uLnRoaXMgaXMgdGhlIHBsYWNlIHRvIGdvIG5vdy4g
 QnV0IHRoZSBza3kga25vd3MgdGhlIHJlYXNvbnMgYW5kIHRo
 ZSBwYXR0ZXJucyBiZWhpbmQgYWxsIGNsb3VkcywgYW5kIHlv
 dSB3aWxsIGtub3csIHRvbywgd2hlbiB5b3UgbGlmdCB5b3Vy
 c2VsZiBoaWdoIGVub3VnaCB0byBzZWUgYmV5b25kIGhvcml6
 b25zLiINCg0KLVJpY2hhcmQgQmFjaA=="
            }
        ],
        "networks": [
            {
                 "uuid": "f212726e-6321-4210-9bae-a13f5a33f83f"
            },
            {
                 "uuid": "00000000-0000-0000-0000-000000000000"
            },
            {
                 "uuid": "11111111-1111-1111-1111-111111111111"
            }
        ]
    }
}
```

Example 6.8. Provision Servers and Attach Networks: XML request

```
<?xml version="1.0" encoding="UTF-8"?>
<server xmlns="http://docs.openstack.org/compute/api/v1.1" imageRef=
"3afe97b2-26dc-49c5-a2cc-a2fc8d80c001" flavorRef="2"
  name="api-test-server-xml2">
  <metadata>
    <meta key="My Server Name">API Test Server XML</meta>
  </metadata>
  <personality>
    <file path="/etc/banner.txt">
 ICAgICAgDQoiQSBjbG91ZCBkb2VzIG5vdCBrbm93IHdoeSBp
      dCBtb3ZlcyBpbiBqdXN0IHN1Y2ggYSBkaXJlY3Rpb24gYW5k
 IGF0IHN1Y2ggYSBzcGVlZC4uLkl0IGZlZWxzIGFuIGltcHVs
      c2lvbi4uLnRoaXMgaXMgdGhlIHBsYWNlIHRvIGdvIG5vdy4g
 QnV0IHRoZSBza3kga25vd3MgdGhlIHJlYXNvbnMgYW5kIHRo
      ZSBwYXR0ZXJucyBiZWhpbmQgYWxsIGNsb3VkcywgYW5kIHlv
 dSB3aWxsIGtub3csIHRvbywgd2hlbiB5b3UgbGlmdCB5b3Vy
      c2VsZiBoaWdoIGVub3VnaCB0byBzZWUgYmV5b25kIGhvcml6
 b25zLiINCg0KLVJpY2hhcmQgQmFjaA==</file>
  </personality>
  <networks>
    <network uuid="0ef47ac7-6797-4e01-8a47-ed26ec3aaa56"/>
    <network uuid="00000000-0000-0000-0000-000000000000"/>
    <network uuid="11111111-1111-1111-1111-111111111111"/>
  </networks>
```

```
  <keypair>
    <key_name>name_of_keypair-96bbe50e-05e1-4d59-9115-4779a3ebcc2e</key_name>
  </keypair>
</server>
```

This operation does not require a request body.

Response

This list shows the body parameters for the response:

- **parameters**:

  - **OS-DCF:diskConfig**: String. Required.

    The disk configuration value, which is AUTO or MANUAL. See Disk Configuration Extension.

  - **adminPass**: String. Required.

    The Admin password for the server.

  - **id**: UUID. Required.

    The ID for the server.

  - **links**: Array. Required.

    The *bookmark* and *self* for the server. For more information on links, see Links and References in the *Next Generation Cloud Servers Developer Guide*.

Example 6.9. Provision Servers and Attach Networks: JSON response

```
{
    "server": {
        "OS-DCF:diskConfig": "AUTO",
        "adminPass": "LMoheHauXt8w",
        "id": "ef08aa7a-b5e4-4bb8-86df-5ac56230f841",
        "links": [
            {
                "href": "https://dfw.servers.api.rackspacecloud.com/v2/010101/
servers/ef08aa7a-b5e4-4bb8-86df-5ac56230f841",
                "rel": "self"
            },
            {
                "href": "https://dfw.servers.api.rackspacecloud.com/010101/
servers/ef08aa7a-b5e4-4bb8-86df-5ac56230f841",
                "rel": "bookmark"
            }
        ]
    }
}
```

Example 6.10. Provision Servers and Attach Networks: XML response

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

```
<server
    xmlns:OS-DCF="http://docs.openstack.org/compute/ext/disk_config/api/v1.1"
    xmlns:atom="http://www.w3.org/2005/Atom"
    xmlns="http://docs.openstack.org/compute/api/v1.1"
    id="ed5c7754-29b6-45fa-96cb-ab64958c8c0a" adminPass="Dd5pNZtpVVQ3"
    OS-DCF:diskConfig="AUTO">
    <metadata/>
    <atom:link
        href="https://dfw.servers.api.rackspacecloud.com/v2/010101/servers/
ed5c7754-29b6-45fa-96cb-ab64958c8c0a"
        rel="self"/>
    <atom:link
        href="https://dfw.servers.api.rackspacecloud.com/010101/servers/
ed5c7754-29b6-45fa-96cb-ab64958c8c0a"
        rel="bookmark"/>
</server>
```

This operation does not return a response body.

Show Network

Method	URI	Description
GET	/os-networksv2/{id}	Shows information for the specified network.

This operation shows information for the network specified in the request URI.

Note

To list the networks that are associated with servers, see List Servers in the *Next Generation Cloud Servers Developer Guide*.

Normal response codes: 200

Error response codes: unauthorized (401), forbidden (403), networkNotFound (420)

Request

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

Name	Type	Description
{id}	UUID	The network ID.

This operation does not require a request body.

Response

This list shows the body parameters for the response:

- **parameters**:
  - **id**: Xsd:string.

    The network ID.
  - **label**: Xsd:string.

    The name of the network.
  - **cidr**: Xsd:string.

    The CIDR for an isolated network.

Example 6.11. Show Network: JSON response

```
{
    "network": {
        "cidr": "192.168.0.0/24",
        "id": "f212726e-6321-4210-9bae-a13f5a33f83f",
        "label": "superprivate_xml"
    }
}
```

Example 6.12. Show Network: XML response

```
<network>
    <cidr>192.168.0.0/24</cidr>
    <id>f212726e-6321-4210-9bae-a13f5a33f83f</id>
    <label>superprivate_xml</label>
</network>
```

This operation does not return a response body.

Delete Network

Method	URI	Description
DELETE	/os-networksv2/{id}	Deletes the specified network.

This operation deletes the network specified in the request URI.

Note

You can delete an isolated network only if it is not attached to any server. To detach a network from a server, delete the virtual interface for the network from the server instance. See the section called "Delete Virtual Interface" [123].

Normal response codes: 202

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

Request

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

Name	Type	Description
{id}	UUID	The network ID.

This operation does not require a request body.

Extension - Cloud Networks Virtual Interfaces

Use the Cloud Networks virtual interface extension to create a virtual interface to a specified network and attach that network to an existing server instance. You can attach either an isolated network that you have created or a Rackspace network.

A virtual interface works in the same way as the network interface card (NIC) inside your laptop. Like a NIC, a virtual interface is the medium through which you can attach a network to your server. You create a virtual interface for a specified network, and the network, which is composed of logical switches, is attached to your server instance through the virtual interface.

Note

You can create a maximum of one virtual interface per instance per network.

You can also use the Cloud Networks virtual interface extension to perform the following actions:

- List the virtual interfaces for a server instance.
- Delete a virtual interface and detach it from a server instance.

Method	URI	Description
GET	`/servers/{instance_id}/os-virtual-interfacesv2`	Lists the virtual interfaces configured for a server instance.
POST	`/servers/{instance_id}/os-virtual-interfacesv2`	Creates a virtual interface for a network and attaches the network to a server instance.
DELETE	`/servers/{instance_id}/os-virtual-interfacesv2/{interface_id}`	Deletes a virtual interface from a server instance.

List Virtual Interfaces

Method	URI	Description
GET	`/servers/{instance_id}/os-virtual-interfacesv2`	Lists the virtual interfaces configured for a server instance.

This operation lists the virtual interfaces configured for a server instance.

Normal response codes: 200

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

Request

This table shows the URI parameters for the list virtual interfaces request:

Name	Type	Description
`{instance_id}`	UUID	The instance ID.

This operation does not require a request body.

Response

This list shows the body parameters for the response:

- **parameters**:
  - **virtual_interfaces**: Array.

    The array of virtual interfaces.
    - **id**: Xsd:string.

      The virtual interface ID.
    - **ip_addresses**: Array.

      The array of interface IP address details.
      - **address**: Xsd:string.

        The interface IP address.
      - **network_id**: Xsd:string.

        The interface network ID.
      - **network_label**: Xsd:string.

        The interface network label.
    - **mac_address**: Xsd:string.

The Media Access Control (MAC) address for the virtual interface. A MAC address is a unique identifier assigned to network interfaces for communications on the physical network segment.

Example 6.13. List Virtual Interfaces: JSON response

```
{
    "virtual_interfaces": [
        {
            "id": "a589b11b-cd51-4274-8ec0-832ce799d156",
            "ip_addresses": [
                {
                    "address": "2001:4800:7810:0512:d87b:9cbc:ff04:850c",
                    "network_id": "ba122b32-dbcc-4c21-836e-b701996baeb3",
                    "network_label": "public"
                },
                {
                    "address": "64.49.226.149",
                    "network_id": "ba122b32-dbcc-4c21-836e-b701996baeb3",
                    "network_label": "public"
                }
            ],
            "mac_address": "BC:76:4E:04:85:0C"
        },
        {
            "id": "de7c6d53-b895-4b4a-963c-517ccb0f0775",
            "ip_addresses": [
                {
                    "address": "192.168.0.2",
                    "network_id": "f212726e-6321-4210-9bae-a13f5a33f83f",
                    "network_label": "superprivate_xml"
                }
            ],
            "mac_address": "BC:76:4E:04:85:20"
        },
        {
            "id": "e14e789d-3b98-44a6-9c2d-c23eb1d1465c",
            "ip_addresses": [
                {
                    "address": "10.181.1.30",
                    "network_id": "3b324a1b-31b8-4db5-9fe5-4a2067f60297",
                    "network_label": "private"
                }
            ],
            "mac_address": "BC:76:4E:04:81:55"
        }
    ]
}
```

Example 6.14. List Virtual Interfaces: XML response

```
<?xml version='1.0' encoding='UTF-8'?>
<virtual_interfaces xmlns="http://docs.openstack.org/compute/api/v1.1">
    <virtual_interface id="a589b11b-cd51-4274-8ec0-832ce799d156" mac_address=
"BC:76:4E:04:85:0C">
        <ip_address network_id="ba122b32-dbcc-4c21-836e-b701996baeb3"
 network_label="public" address="2001:4800:7810:0512:d87b:9cbc:ff04:850c" />
        <ip_address network_id="ba122b32-dbcc-4c21-836e-b701996baeb3"
 network_label="public" address="64.49.226.149" />
```

```
    </virtual_interface>
    <virtual_interface id="de7c6d53-b895-4b4a-963c-517ccb0f0775" mac_address=
"BC:76:4E:04:85:20">
        <ip_address network_id="f212726e-6321-4210-9bae-a13f5a33f83f"
 network_label="superprivate_xml" address="192.168.0.2" />
    </virtual_interface>
    <virtual_interface id="e14e789d-3b98-44a6-9c2d-c23eb1d1465c" mac_address=
"BC:76:4E:04:81:55">
        <ip_address network_id="3b324a1b-31b8-4db5-9fe5-4a2067f60297"
 network_label="public" address="10.181.1.30" />
    </virtual_interface>
</virtual_interfaces>
```

This operation does not return a response body.

Create Virtual Interface

Method	URI	Description
POST	/servers/{instance_id}/os-virtual-interfacesv2	Creates a virtual interface for a network and attaches the network to a server instance.

This operation creates a virtual interface for a network and attaches the network to a server instance.

Note

You can create a maximum of one virtual interface per instance per network.

Normal response codes: 200

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

Request

This table shows the URI parameters for the create virtual interface request:

Name	Type	Description
{instance_id}	UUID	The instance ID.

This list shows the body parameters for the request:

- **parameters**:
  - **network_id**: Xsd:UUID. Required.

    The ID of the network for which you want to create a virtual interface. You can create a virtual interface for an isolated network or for the Rackspace networks.

Example 6.15. Create Virtual Interface: JSON request

```
{
   "virtual_interface":
    {
      "network_id": "1f7920d3-0e63-4fec-a1cb-f7916671e8eb"
    }
}
```

Example 6.16. Create Virtual Interface: XML request

```
<?xml version="1.0" encoding="UTF-8"?>
<virtual_interface
  network_id="0ef47ac7-6797-4e01-8a47-ed26ec3aaa56"
/>
```

This operation does not require a request body.

Response

This list shows the body parameters for the response:

- **parameters**:

  - **virtual_interfaces**: Array.

    The array of virtual interfaces.

    - **id**: Xsd:string.

      The virtual interface ID.

    - **ip_addresses**: Array.

      The array of interface IP address details.

      - **address**: Xsd:string.

        The interface IP address.

      - **network_id**: Xsd:string.

        The interface network ID.

      - **network_label**: Xsd:string.

        The interface network label.

    - **mac_address**: Xsd:string.

      The Media Access Control (MAC) address for the virtual interface. A MAC address is a unique identifier assigned to network interfaces for communications on the physical network segment.

Example 6.17. Create Virtual Interface: JSON response

```
{
   "virtual_interfaces":[
      {
         "mac_address":"FE:ED:FA:00:08:93",
         "id":"045f195f-3347-487b-8e80-8ee3390eda56",
         "ip_addresses":[
            {
               "address":"192.168.0.1",
               "network_id":"196a0246-86cc-46fa-9ecf-850f67c2cb7c",
               "network_label":"added_network"
            }
         ]
      }
   ]
}
```

Example 6.18. Create Virtual Interface: XML response

```
<?xml version='1.0' encoding='UTF-8'?>
<virtual_interfaces xmlns="http://docs.openstack.org/compute/api/v1.1">
    <virtual_interface id="24293921-e8fe-4f93-ac52-b8cc08435d00"
        mac_address="FE:ED:FA:00:0D:13">
```

```
        <ip_address address="1.1.1.129"
            network_id="6d17d84a-9513-4c4c-bf5a-c2d5c0794292"
            network_label="added_network"/>
    </virtual_interface>
</virtual_interfaces>
```

This operation does not return a response body.

Delete Virtual Interface

Method	URI	Description
DELETE	`/servers/{instance_id}/os-virtual-interfacesv2/{interface_id}`	Deletes a virtual interface from a server instance.

This operation deletes the specified virtual interface from the specified server instance.

To find the ID of a virtual interface, issue the List Virtual Interfaces API operation. See the section called "List Virtual Interfaces" [117].

Normal response codes: 200

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

Request

This table shows the URI parameters for the delete virtual interface request:

Name	Type	Description
`{instance_id}`	UUID	The instance ID.
`{interface_id}`	UUID	The interface ID.

This operation does not require a request body.