

OpenStack Networking API in CEE in HDS Deployment

Cloud Execution Environment

INTERWORK DESCRIPTION

Copyright

© Ericsson AB 2017, 2018. All rights reserved. No part of this document may be reproduced in any form without the written permission of the copyright owner.

Disclaimer

The contents of this document are subject to revision without notice due to continued progress in methodology, design and manufacturing. Ericsson shall have no liability for any error or damage of any kind resulting from the use of this document.

Trademark List

All trademarks mentioned herein are the property of their respective owners. These are shown in the document Trademark Information.



Contents

1	Introduction	1
1.1	API Version	1
1.2	Document References	1
2	Prerequisites	2
2.1	CEE Networking Configurations	2
2.2	Segmentation IDs	2
3	Supported Operations	3
3.1	Basic OpenStack Operations	3
3.2	OpenStack Extensions	3
4	Deviations	4
4.1	Partly Supported Operations	4
4.2	Added Operations	4
4.3	Operations Not Available in CEE	4
5	Ericsson Extensions	6
5.1	Trunkport Extension	6
6	Limitations and Recommendations	14
6.1	Limitations	14
6.2	Recommendations	15
7	Concepts and Use Cases	16
7.1	General Terms	16
7.2	IP Address Management	16
7.3	MAC Address Management	17
7.4	Internal Neutron Network without Neutron IP Address Management	17
7.5	Internal Neutron Network with Neutron IP Address Management	18
7.6	Networks for CEE Region External Connectivity	18





1 Introduction

This document describes the use of the Application Programming Interface (API) for networking in the Cloud Execution Environment (CEE) on Hyperscale Datacenter System (HDS) without SDN component. The API is based on the OpenStack networking component (Neutron).

1.1 API Version

The CEE Networking API is based on OpenStack Networking API v2.

1.2 Document References

This section contains the official OpenStack API reference.

1.2.1 API Design Base Reference

For the description of the API operations and extensions of networking, refer to the [OpenStack Networking API v2.0](#).

This is a stored copy of the OpenStack API Reference document version that was the base for the development of this version of CEE.

Note: It is possible that the date on the front page differs from the revision date in this document. The front page shows the date on which the document was generated.



2 Prerequisites

The following sections list the prerequisites.

2.1 CEE Networking Configurations

CEE networking can be configured during the installation of CEE. For details about the configurations, refer to the [Configuration File Guide](#).

Table 1 Neutron Configurations

Name	Neutron in CEE, Compared to Standard OpenStack Neutron
ericsson_user_spec	<ul style="list-style-type: none">• Does not use L3 agent and floating IPs.• Uses several extensions that are explained in this document.• Can use additional features if properly configured by the user at installation of CEE• Does not use L3 service plugin.

2.2 Segmentation IDs

A segmentation ID is implemented as a VLAN tag. The range must be defined during the installation of CEE. CEE region external connectivity also requires a range of VLANs to be specified at installation of CEE. The cloud administrator has to use the latter range when using the Neutron provider extension, as described in the [OpenStack Networking API v2.0](#). The two ranges must not overlap.

Table 2 VLAN Allocation Example

Name	Range	Description
Default	130-3999	Used by Neutron for tenant separation
Provider	50-129	Used for CEE region external connectivity

Note: The table is an example, it does not mandate specific or default values. The values must be configured before the CEE region is installed. The configuration of values is described in the [Configuration File Guide](#).

Each Neutron network requires one segmentation ID. Segmentation IDs have to be unique, they cannot be used on several Neutron networks. Segmentation IDs on Neutron networks are static and cannot be changed after creation of the Neutron network.



3 Supported Operations

The following sections contain information about the API operations and API extensions in CEE.

3.1 Basic OpenStack Operations

For the detailed description of basic Networking API operations, refer to the [OpenStack Networking API v2.0](#).

3.1.1 Limitations

For the CEE-specific limitations and recommendations, see Section 6 on page 14.

3.2 OpenStack Extensions

This section contains information about which Networking API extensions are supported.

- Agent Management Extension

The following agents are supported:

- neutron-dhcp-agent
- neutron-openvswitch-agent

- Agent Schedulers

The following agents are supported:

- DHCP Agent Scheduler
- Device Agent Scheduler

- List Extensions

Note: Supported, but the reply contains unsupported operations.

- Networks multiple provider extension (networks)
- Quotas extension (quotas)
- Network provider extended attributes (networks)
- Ports binding extended attributes (ports)



4 Deviations

This section describes the deviations between vanilla OpenStack networking and CEE networking.

4.1 Partly Supported Operations

The following operations are partly supported in CEE networking:

Operations for subnets

Only IPv4 is supported.

Administrative State Down (ports)

Administration state must be up for trunk ports and their subports.

If the administration state of a trunk port is set to down at the CIC, it will not take effect to the trunk port and its subports at the relevant Compute node, although Neutron will show that the trunk port and its subports are administratively down.

4.2 Added Operations

The following extension is added:

— Trunkport Extension

The Neutron Trunkport extension allows Virtual Machines (VMs) to send VLAN tagged traffic to Neutron networks. In this way, each VLAN tag is associated with a Neutron network.

4.3 Operations Not Available in CEE

The following operations are not available in CEE Networking:

- Configurable external gateway modes
- Extra DHCP options (extra-dhcp-opt)
- The ExtraRoute Extension
- Firewall as a Service (FaaS)
- Load Balancer as a Service (LBaaS)
- Metering labels and rules



- Security groups and rules (`security-groups`)

Note: Although the Neutron security groups API is operational, functionality depends on the FW driver.

- Virtual Private Network as a Service (VPNaaS)
- Multiple vNICs in the same network.
- Neutron routers
- Floating IP



5 Ericsson Extensions

This section describes the added CEE API extensions in detail.

5.1 Trunkport Extension

This section describes the added CEE API extensions in detail.

Note: The trunkport extension is deprecated and will be removed in the next CEE release.

The Neutron Trunkport extension allows VMs to send VLAN tagged traffic to Neutron networks in a way that each VLAN tag is associated with a Neutron network.

5.1.1 Topology

A trunkport can be a simple trunkport or a subport.

A trunkport is a port that is connected to the VMs (representing a Network Interface Card - NIC).

Trunkports function much like a normal Neutron port.

For outgoing traffic from a VM, untagged traffic is forwarded to the network to which the trunkport is connected.

Incoming traffic from trunkports is untagged when entering the VM.

A subport is a port that is on a trunkport, and connected to a Neutron network.

For outgoing traffic from a VM, the VLAN tag is stripped off before the traffic is forwarded to the network connected to a subport. The VLAN tag from the VM (the one associated with the subport) is stripped before entering the associated Neutron network. When entering the associated Neutron network, the VLAN tag associated with the Neutron network is added.

For incoming traffic to a VM, from the network connected to a subport, the VLAN tag associated with the Neutron network is stripped of and replaced with the VLAN tag associated with the subport, before the traffic is forwarded to a VM.

The topology of the trunkport concept is shown in Figure 1.

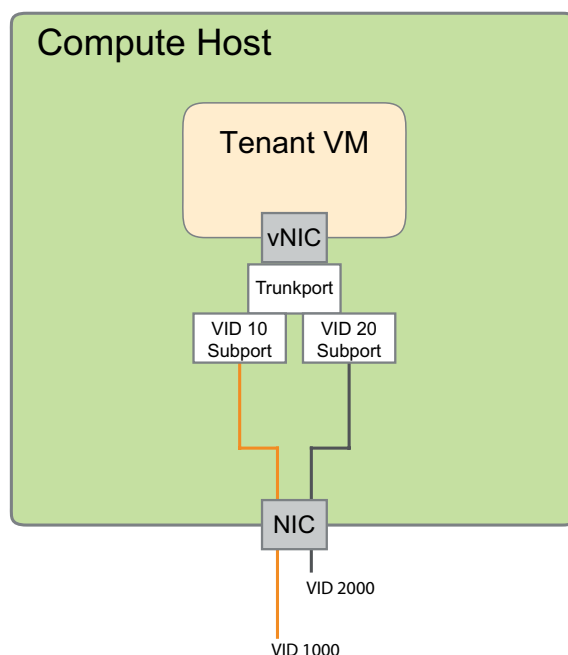


Figure 1 Trunkport Topology

5.1.2 Concepts

The `trunkport-`prefixed extended attributes for the **port** resource are shown in Table 3.

Table 3 Trunkport Attributes

Attribute	Type	Required	CRUD ⁽¹⁾	Default Value	Validation Constraints	Notes
<code>trunkport:type</code>	String	N/A	CR	None	Trunkport or subport	Defines whether a port is a trunkport or a subport.
<code>trunkport:parent_id</code>	uuid-string	N/A	CR	None	Valid port id of a trunkport	For subports, the id of the trunkport to which the subport is connected.
					Unset	For trunkports, unset.



Table 3 Trunkport Attributes

Attribute	Type	Required	CRUD ⁽¹⁾	Default Value	Validation Constraints	Notes
trunkport:vid	Integer	N/A	CR	None	1-4094	For subports, segmentation ID that is used to access the given subport from the trunkport.
					Unset	For trunkports, unset.

(1) 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.

5.1.3 Trunkport API Operations

This section discusses the operations for setting and retrieving the trunkport port extension attributes for port objects.

5.1.3.1 List Ports

Table 4 List Ports

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

Normal response code: 200

Error response code: Unauthorized (401)

This operation returns all the ports defined in Neutron to which the given user has access.

```
{
  "ports": [
    {
      "status": "DOWN",
      "binding:host_id": null,
      "name": "",
      "allowed_address_pairs": [],
      "admin_state_up": true,
      "network_id": "87733bcc-8144-41b1-bb6b-d011d7a5168e",
      "tenant_id": "7ea98790cd854fb5a82ef3d41e5c156b",
      "extra_dhcp_opts": [{"opt_value": "testfile.1", =>
        "opt_name": "bootfile-name"}],
    }
  ]
}
```



```
{
  "opt_value": "123.123.123.45", "opt_name": "server-ip-address",
  "opt_value": "123.123.123", "opt_name": "tftp-server"
}, {
  "binding:vif_type": "ovs",
  "device_owner": "",
  "binding:capabilities": {"port_filter": true},
  "mac_address": "fa:16:3e:52:92:3a",
  "fixed_ips": [{"subnet_id": "99a8aea3-b9da-409d-a5e5-f45338ceb4d3",
    "ip_address": "172.24.4.228"}],
  "id": "3c0c7a37-690a-43a8-8088-5d4c2c7f8484",
  "security_groups": ["9bf6f19a-ba4a-470f-b8ce-28c9ad66556c"],
  "device_id": "",
  "trunkport:type": "trunk",
  "trunkport:parent_id": "",
  "trunkport:vid": ""
}, {
  "status": "ACTIVE",
  "binding:host_id": null,
  "name": "",
  "allowed_address_pairs": [],
  "admin_state_up": true,
  "network_id": "87733bcc-8144-41b1-bb6b-d011d7a5168e",
  "tenant_id": "7ea98790cd854fb5a82ef3d41e5c156b",
  "extra_dhcp_opts": [],
  "binding:vif_type": "ovs",
  "device_owner": "compute:probe",
  "binding:capabilities": {"port_filter": true},
  "mac_address": "fa:16:3e:49:56:07",
  "fixed_ips": [{"subnet_id": "99a8aea3-b9da-409d-a5e5-f45338ceb4d3",
    "ip_address": "172.24.4.227"}],
  "id": "5212d40a-c2f5-4a5d-ad18-694658047654",
  "security_groups": ["9bf6f19a-ba4a-470f-b8ce-28c9ad66556c"],
  "device_id": "zvm2",
  "trunkport:type": "subport",
  "trunkport:parent_id": "3c0c7a37-690a-43a8-8088-5d4c2c7f8484",
  "trunkport:vid": "10"
}
]
```

Example 1 List Ports: JSON Response



5.1.3.2 Show Port

Table 5 Show Port

Verb	URI	Description
GET	/ports/port_id	Returns details about a specific port including trunkport attributes.

Normal response code: 200

Error response codes: Unauthorized (401), Not Found (404)

This operation returns the port attributes of a port specified in the request URI. These attributes include the trunkport attributes.

```
{
  "port":
  {
    "status": "DOWN",
    "binding:host_id": null,
    "name": "",
    "allowed_address_pairs": [],
    "admin_state_up": true,
    "network_id": "87733bcc-8144-41b1-bb6b-d011d7a5168e",
    "tenant_id": "7ea98790cd854fb5a82ef3d41e5c156b",
    "extra_dhcp_opts": [
      {"opt_value": "testfile.1", "opt_name": "bootfile-name"},
      {"opt_value": "123.123.123.123", "opt_name": "tftp-server"},
      {"opt_value": "123.123.123.45", "opt_name": "server-ip-address"}
    ],
    "binding:vif_type": "ovs",
    "device_owner": "",
    "binding:capabilities": {"port_filter": true},
    "mac_address": "fa:16:3e:52:92:3a",
    "fixed_ips": [{"subnet_id": "99a8aea3-b9da-409d-a5e5-f45338ceb4d3",
      "ip_address": "172.24.4.228"}],
    "id": "3c0c7a37-690a-43a8-8088-5d4c2c7f8484",
    "security_groups": ["9bf6f19a-ba4a-470f-b8ce-28c9ad66556c"],
    "device_id": "",
    "trunkport:type": "trunk",
    "trunkport:parent_id": "",
    "trunkport:vid": ""
  }
}
```

Example 2 Show Port with Trunkport Attributes: JSON Response



5.1.3.3 Create Port

Table 6 Create Port

Verb	URI	Description
POST	/ports	The operation creates a new port, and the supplied attributes show whether the port is a trunkport or a subport.

Normal response code: 200

Error response code: Unauthorized (401)

The operation creates a new port, and the supplied attributes show whether the port is a trunkport or a subport.

Note: When creating a subport, `mac_address` is copied from the parent port.

```
{
  "port":
  {
    "network_id": "87733bcc-8144-41b1-bb6b-d011d7a5168e",
    "fixed_ips": [{"subnet_id": "99a8aea3-b9da-409d-a5e5-f45338ceb4d3",
      "ip_address": "172.24.4.230"}],
    "admin_state_up": true,
    "trunkport:type": "trunk"
  }
}
```

Example 3 Create Port with Trunkport Attributes: JSON Request



```
{
  "port":
  {
    "status": "DOWN",
    "binding:host_id": null,
    "name": "",
    "allowed_address_pairs": [],
    "admin_state_up": true,
    "network_id": "87733bcc-8144-41b1-bb6b-d011d7a5168e",
    "tenant_id": "7ea98790cd854fb5a82ef3d41e5c156b",
    "binding:vif_type": "ovs",
    "device_owner": "",
    "binding:capabilities": {"port_filter": true},
    "mac_address": "fa:16:3e:43:3c:b7",
    "fixed_ips": [{"subnet_id":
    : "99a8aea3-b9da-409d-a5e5-f45338ceb4d3",
    "ip_address": "172.24.4.230"}],
    "id": "055d27c0-0194-4782-be45-275ff2c95c61",
    "security_groups": ["9bf6f19a-ba4a-470f-b8ce-28c9ad66556c"],
    "device_id": "",
    "trunkport:type": "trunk",
    "trunkport:parent_id": "",
    "trunkport:vid": ""
  }
}
```

Example 4 Create Port with Trunkport Attributes: JSON Response

```
{
  "port":
  {
    "network_id": "87733bcc-8144-41b1-bb6b-d011d7a5168e",
    "fixed_ips": [{"subnet_id":
    : "99a8aea3-b9da-409d-a5e5-f45338ceb4d3",
    "ip_address": "172.24.4.230"}],
    "admin_state_up": true,
    "trunkport:type": "subport",
    "trunkport:parent_id": "3c0c7a37-690a-43a8-8088-5d4c2c7f8484",
    "trunkport:vid": "10"
  }
}
```

Example 5 Create Port with Trunkport Attributes (Type Subport): JSON Request



```
{
  "port":
  {
    "status": "DOWN",
    "binding:host_id": null,
    "name": "",
    "allowed_address_pairs": [],
    "admin_state_up": true,
    "network_id": "87733bcc-8144-41b1-bb6b-d011d7a5168e",
    "tenant_id": "7ea98790cd854fb5a82ef3d41e5c156b",
    "binding:vif_type": "ovs",
    "device_owner": "",
    "binding:capabilities": {"port_filter": true},
    "mac_address": "fa:16:3e:43:3c:b7",
    "fixed_ips": [{"subnet_id":
      : "99a8aea3-b9da-409d-a5e5-f45338ceb4d3",
    "ip_address": "172.24.4.230"}],
    "id": "055d27c0-0194-4782-be45-275ff2c95c61",
    "security_groups": ["9bf6f19a-ba4a-470f-b8ce-28c9ad66556c"],
    "device_id": "",
    "trunkport:type": "subport",
    "trunkport:parent_id": "3c0c7a37-690a-43a8-8088-5d4c2c7f8484",
    "trunkport:vid": "10"
  }
}
```

Example 6 Create Port with Trunkport Attributes (Type Subport): JSON Response

5.1.3.4 Update Port

No attributes can be updated.

5.1.3.5 Delete Port

When deleting a trunkport, the subports that are connected to it are also deleted.



6 Limitations and Recommendations

This section lists the limitations and recommendations for CEE Networking.

6.1 Limitations

This section contains the limitations of CEE Networking.

6.1.1 Segmentation IDs

The total number of segmentation IDs is limited to 4094, see Section 2.2 on page 2.

6.1.2 Neutron Network Memory Allocation

Each Neutron network created consumes RAM in the vCIC, and it influences the maximum number of virtual tenant networks. Refer to the [Multi-Server System Dimensioning Guide, CEE 6](#) for more information.

6.1.3 Deleting Trunkport and Subports

If a trunk port is deleted before the associated subports have been deleted, there are remains left in Neutron that prevent using the same IP address with a different MAC number. If a new trunk port and subports are created with the same IP address, this can prevent VMs on the ports to get an IP address.

To delete a trunk port with subports, do the following:

1. Delete the related subports.
2. Delete the trunk port.

6.1.4 ARP Anti-Spoofing and Allowed Address Pairs for Trunkports and Subports

For trunkports and subports, ARP anti-spoofing and allowed address pairs are not available.

6.1.5 Updating port_security_enabled Flag

If the Neutron port parameter `port_security_enabled` is modified with `neutron port-update` command after port creation, the effect does not cascade down to the port.

To modify the value of the flag and expect the behavior change, do the following:



1. Delete the port.
2. Re-create the port with the required `port_security_enabled` flag.

6.1.6 VFs Do Not Obtain the IP Addresses Assigned to Them by the Neutron DHCP Service

The provisioned VMs which have SR-IOV VF interfaces with `flat` network type cannot use the Neutron DHCP service. Therefore, they must be configured manually in the guest VM.

6.1.7 VLAN Tagging Is Not Removed for VFs Belonging to a Deleted Port or Network

Neutron does not remove the VLAN tagging from a VF belonging to an already deleted port or network if the VF interface is not in use. However, the VF is free for other use, and once assigned to a VM, the VLAN ID and the MAC address will change accordingly.

6.1.8 IPv6

Network Routing or Address Management for IPv6 is not available.

6.2 Recommendations

There are no recommendations for HDS.



7 Concepts and Use Cases

The following sections describe the various concepts and use cases that are connected to the CEE Networking API.

7.1 General Terms

Neutron Network

L2 broadcast domain

Neutron Subnet

Definition of a subnet. It s attached to a Neutron network. Used to configure the Neutron DHCP service.

7.2 IP Address Management

The following three methods are available for IP address management:

- Application handles the IP addresses on its own and do not involve Neutron.

No Neutron DHCP service is used. Only L2 services from Neutron can be used.

Note: Nova requires a subnet attached to a Neutron network in order to start VMs on the network. User is required to create a dummy subnet with DHCP disabled as a workaround.

- Application uses Neutron for IP address management.

This can be done in several ways: the application can specify IP addresses per ports or only specify IP subnets and let Neutron allocate the addresses.

- A mix of the above two.

In this case, it is the responsibility of the application to align the configuration view of the application with the configuration view of Neutron.

IP addresses in Neutron can be reused on different Neutron networks. IP addresses known by Neutron must have matching Neutron subnets.

To manually assign an IP address to a port use the `fixed_ips` attribute in the Neutron port when creating the Neutron port.

The `fixed_ips` attribute includes `ip_address` and `subnet_id`. Refer to the OpenStack Networking API v2.0.

7.2.1 Neutron DHCP

Neutron DHCP can be enabled or disabled per subnet.



This is done by using the `enable_dhcp` attribute on the Neutron subnet.

`enable_dhcp <True/False>`

7.3 MAC Address Management

The following two methods are available for MAC address management:

- The application handles MAC addresses on its own, and provides the MAC address to Neutron when creating the Neutron port.
- The application allows Neutron to allocate a MAC address when creating the Neutron port.

MAC addresses have to be unique per Neutron network, but can be reused on different Neutron networks.

To manually assign a MAC address use the `mac_address` attribute in the Neutron port when creating the Neutron port, refer to the [OpenStack Networking API v2.0](#).

Neutron-allocated MAC addresses have a three byte fixed prefix. The rest will be a random number unique per Neutron network. Refer to the [Configuration File Guide](#) for the prefix used in the Neutron configuration file.

Note: It is advisable to use the local administered MAC ranges.

7.4 Internal Neutron Network without Neutron IP Address Management

This section describes how to connect VMs using internal L2. Create Neutron network, create Neutron ports on Neutron network, and then create VMs using those Neutron ports.

1. Create Neutron network.

For the procedure of creating a Neutron network, refer to the [OpenStack Networking API v2.0](#).

2. Create Neutron subnet on Neutron network.

Nova requires a subnet attached to a Neutron network in order to start VMs on the network. User is required to create a dummy subnet with DHCP disabled as a workaround.

For the procedure of creating a Neutron subnet on Neutron network, refer to the [OpenStack Networking API v2.0](#).

3. Create Neutron ports on Neutron network.

For the procedure of creating Neutron ports on a Neutron network, refer to the [OpenStack Networking API v2.0](#).

4. Create VMs using Neutron ports.

7.5 Internal Neutron Network with Neutron IP Address Management

This section describes how to connect VMs using internal L3.

1. Create Neutron network.

For the procedure of creating a Neutron network, refer to the [OpenStack Networking API v2.0](#).

2. Create Neutron subnet on Neutron network.

For the procedure of creating a Neutron subnet on Neutron network, refer to the [OpenStack Networking API v2.0](#).

3. Create Neutron ports on Neutron network.

For the procedure of creating Neutron ports on a Neutron network, refer to the [OpenStack Networking API v2.0](#).

4. Create VMs using Neutron ports.

7.6 Networks for CEE Region External Connectivity

The following subsections describe how to connect VMs to external DC-GW by using an L2 network.

The L2 DC-GW connection is shown in Figure 2

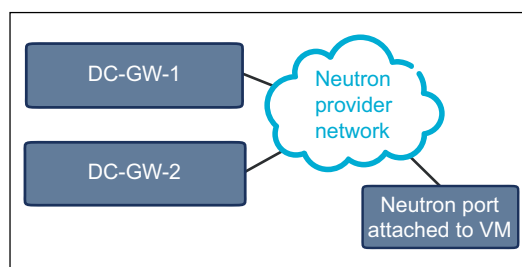


Figure 2 L2 DC-GW Connection

7.6.1 Create Neutron Network

Create a Neutron network using the `provider` network extension (provider segmentation ID). The cloud administrator is required for the management of these IDs. The IDs must be picked from the provider DC-GW pool, see Section 2.2 on page 2.



For the procedure of creating a Neutron network, refer to the [OpenStack Networking API v2.0](#).

For the provider Extended Attributes for Networks, refer to the section [Extensions](#) in the [OpenStack Networking API v2.0](#).

<code>provider:segmentation_id</code>	<code><segmentation_id></code>
<code>provider:network_type</code>	<code>vlan</code>
<code>provider:physical_network</code>	<code>default</code>

7.6.2 Create Neutron Subnet

Create Neutron subnet on Neutron network. A subnet is required regardless if IP address management is going to be handled by Neutron or not.

nova requires a subnet attached to a Neutron network in order to start VMs on the network. User is required to create a dummy subnet with DHCP disabled as a workaround in cases when IP address management is handled by the application.

For the procedure of creating a Neutron subnet on Neutron network, refer to the [OpenStack Networking API v2.0](#).