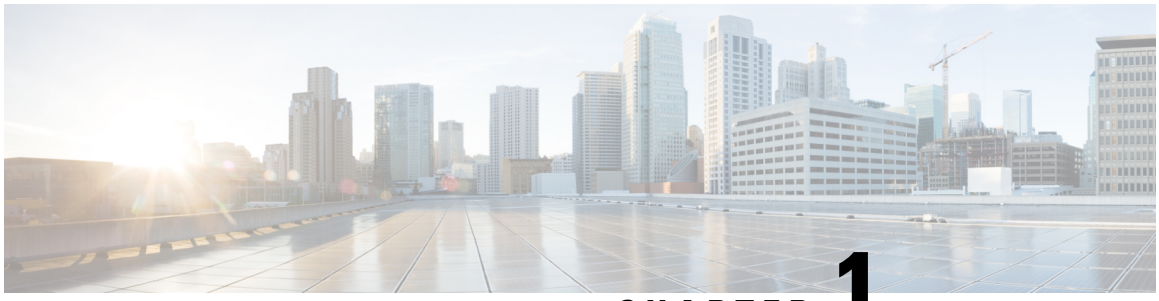
Cisco ACI OpenStack Plugin User Guide

First Published: 2018-12-21

Last Modified: 2020-11-17

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS



CHAPTER 1

New and Changed Information

This chapter contains the following sections:

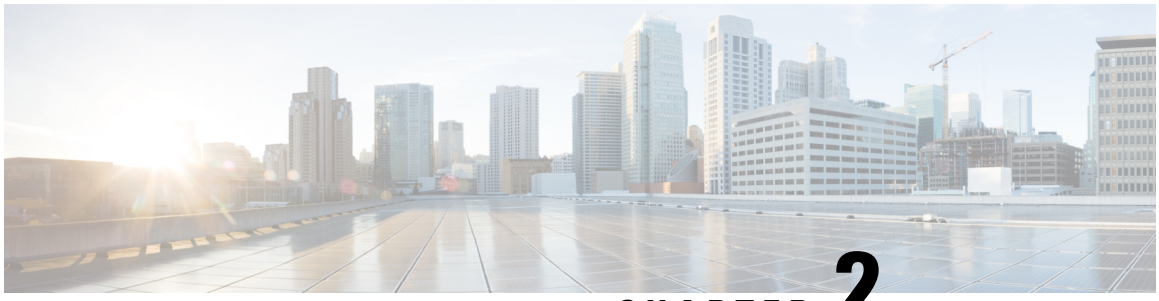
New and Changed Information

The following table provides an overview of the significant changes to this guide up to this current release. The table does not provide an exhaustive list of all changes that are made to the guide or of the new features up to this release.

Table 1: New Features and Changed Behavior

Cisco APIC Release Version	Feature	Description	Where Documented
5.1(1)	Support for dual-stack connections for (SVI) on OpenStack.	You can now configure connections using IPv6 as well as connections using IPv4. This feature enables Border Gateway Protocol (BGP) peering over Layer 3 outside connections for both IPv6 and IPv4 routes if BGP is enabled.	The section Dual Stack with SVI Integration, on page 17 in the chapter "Neutron SVI Integration."
3.0(1)	--	This guide was released.	--
--	--	Added direction for creating a dummy security port group when changing DNS parameters or static routes.	The section Changing DNS Parameters or Static Routes, on page 32 in the chapter "Day 2 Operations."

New and Changed Information



CHAPTER 2

Group-Based Policy

This chapter contains the following sections:

About Group-Based Policy

Group-Based Policy (GBP) is an API framework for OpenStack that offers an intent-driven model intended to describe application requirements in a way that is independent of the underlying infrastructure. Rather than offering network-centric constructs, such as Layer 2 domains, GBP introduces a generic "Group" primitive along with a policy model to describe connectivity, security, and network services between groups. While GBP has focused on the networking domain, it can be a generic framework that extends beyond networking.

GBP runs as a service plug-in within the Neutron process space.

Group-Based Policy Use Cases

Group-Based Policy (GBP) was designed to offer a powerful, but simple language for capturing the requirements of and deploying complex applications on OpenStack clouds. GBP addresses disconnect between application developers who understand application requirements and infrastructure teams who understand various infrastructure capabilities.

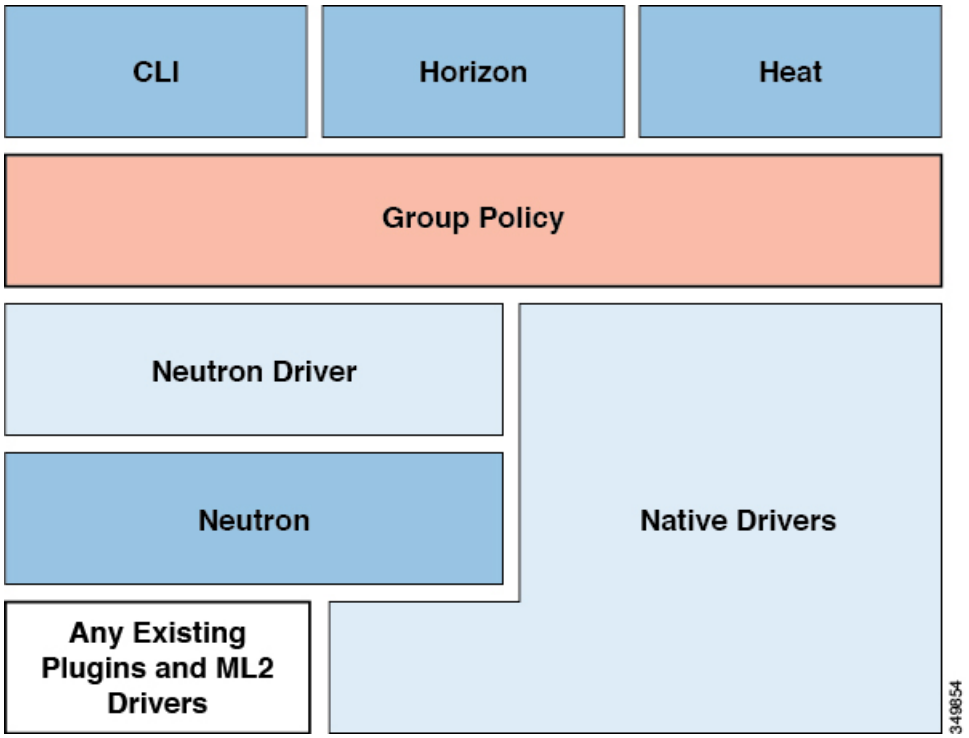
GBP offers the following capabilities beyond what is present in OpenStack:

Dependency mapping	GBP enables users to specify the relationships between different tiers of applications. This dependency map acts as documentation for the security requirements of the application and allows different tiers of the application to evolve separately. The dependency map also makes it extremely easy to scale and automate infrastructure.
Separation of concerns	GBP was designed to separate application security requirements, such as who can talk to whom, from network-specific requirements, such as what IP address ranges to use, where to draw network boundaries, and how to assign virtual IP addresses. This allows application, security, and operation teams to operate independently, but cooperatively.

How Group-Based Policy works

Group-Based Policy (GBP) offers a policy API through multiple OpenStack interfaces, including Horizon extensions (openstack-dashboard-gbp), Heat (openstack-heat-gbp), and cli (openstack-neutron-gbp). GBP was designed to act as a layer on top of Neutron.

GBP supports two forms of mapping to the underlying infrastructure:

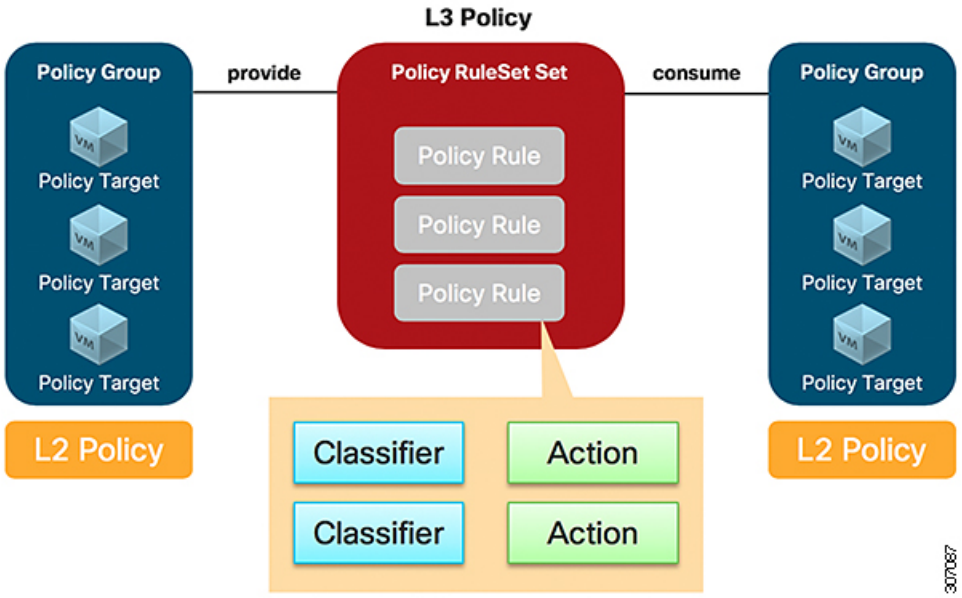


Neutron Mapping Driver	The Neutron mapping driver converts GBP resources into existing Neutron API calls. This allows Neutron to run any existing open source or vendor plugin, including ML2. It also allows GBP to be used in any OpenStack environment. You should define the OpenStack projects either with Neutron API or GBP API. Different projects of the same OpenStack installation could potentially use different APIs. For example, Neutron or GBP.
Native Drivers	You can create drivers that directly render policy constructs through a separate SDN controller or external entity without first converting them to Neutron APIs. This is valuable as it gives the controller additional flexibility on how to interpret and enforce policy without being tied to L2/L3 behaviors. There are currently four native drivers, including Cisco APIC, VMware NSX policy, and One Convergence NSP.

About the Group-Based Policy Model

Group-Based Policy (GBP) introduces a policy model to describe the relationships between different logical groups or tiers of an application. The primitives have been chosen in a way that separates their semantics from the underlying infrastructure capabilities. Resources can be public or local to a specific tenant.

The key primitives are:



Resource	Description
Policy Target	An individual network endpoint (generally a NIC). A policy target is a basic addressable unit in the architecture.
Policy Group	Policy targets with the same properties are organized into policy groups, which is the fundamental primitive of GBP. Policy groups offer an infrastructure agnostic grouping construct without specifying any network semantics in the same way as a broadcast. Each group models its dependencies by declaring the rule sets that it provides to groups as well as rule sets that it will consume.
Policy Classifier	A means of filtering network traffic, including protocol, port range, and direction (in, out, or bidirectional).
Policy Action	An action to take when a particular rule is applied. The supported type include "allow".
Policy Rules	Classifiers paired with actions.
Policy Rule Sets	Policy rule sets contain a number of policy rules. Rule sets can be nested through parent-child relationships.

About Network Policies

Group-Based Policy (GBP) aims to centralize the description of network policies and keep them separate from application-level policies, such as groups and rule sets. This allows a separation of concerns between application owners and cloud/infrastructure administrators.

Resource	Description
Layer 2 Policy	Specifies set of groups within the same switching domain. Layer 2 policies must reference a particular Layer 3 policy.

Resource	Description
Layer 3 Policy	Specifies potentially overlapping IP address space that contains any number of Layer 2 policies.
Network Service Policy	Specifies network-specific parameters that are required for network service chaining, such as VIP allocation.

About Shared Policies

Resources in the Group-Based Policy (GBP) model support a `shared` attribute that can be set to make a resource visible across tenants. The sharing of policy resources promotes the separation of concerns by allowing the relevant team to create a policy and for other users to consume the policy. A shared resource has global visibility, meaning that all tenants are able to see a shared resource. The `shared` attribute can be set or updated only by a user in the administrator role.

The following example creates a shared Layer 3 policy using the CLI:

```
# gbp l3policy-create --shared True my-shared-l3p
```

Similarly, the GUI has a checkbox that enables the sharing or unsharing a resource.

There are certain constraints on when and how a resource is shared:

- A shared resource can only be associated with other shared resources. For example, a shared Layer 2 policy can only exist on a shared Layer 3 policy.
- A shared resource cannot be unshared if it is currently being used.

Note A policy target resource does not support the `shared` attribute.

About the Neutron Mapping Driver

One of the most useful aspects of the Group-Based Policy (GBP) model and its implementation is the ability to map a policy directly into the Neutron API and thus be able to use the existing Neutron plugins as is. The default mapping is as follows:

GBP Resource	Neutron
Policy Target	Port
Policy Target Group	Subnet
Layer 2 Policy	Network
Layer 3 Policy	Router

Note You can design a custom mapping and implement it in a "resource mapping" policy driver.

External Connectivity

The Group-Based Policy (GBP) model supports an API that captures the user's intent to allow the policy targets to communicate with the external world. The following primitives are used to achieve this:

Resource	Description
External Segment	Models an external network that is defined by a Classless Inter-Domain Routing (CIDR) and any other networks that are reachable through this segment using a specified next-hop.
External Policy	Models a policy target group on a external segment. Think of this as an external policy target group that can provide and consume the policy rule set, but the difference is that no policy targets can be created on the external policy.
Neutron Mapping External Segment	The Neutron subnet that is created on a Neutron external network.

This example represents a configuration for an external network with the name "Datacenter-Out". By creating an external segment with the same name, you can automatically configure it with the relevant CIDR and other attributes.

The following command creates an external segment with the name "Datacenter-Out":

```
# export EXT_SEG_ID=$(gbp external-segment-create Datacenter-Out --shared
  True --external-route destination=0.0.0.0/0,nexthop= | awk "/ id / {print \$4}")
```

The following command creates an external policy that models a group that represents the external world:

```
# gbp external-policy-create my-external-policy --external-segments $EXT_SEG_ID
```

The following commands create a policy rule set to allow TCP access:

```
# gbp policy-classifier-create all-tcp-traffic --protocol tcp --direction in
# gbp policy-rule-create tcp-policy-rule --classifier all-tcp-traffic --actions allow
# gbp policy-rule-set-create   tcp-ruleset --policy-rules tcp-policy-rule
```

The following commands set up provide/consume relationships to allow all TCP traffic for the web policy target group to access the external world:

```
# gbp external-policy-update my-external-policy
  --provided-policy-rule-sets "tcp-ruleset=true"
```

This example assumes that the Layer 3 policy is called "default".

The following commands link the implicitly created Layer 3 policy for the web policy target group to the external segment that you created.

```
# gbp l3policy-update default -external-segment Datacenter-Out
```

Network Address Translation Pools and Floating IP Addresses

Each external segment can be configured with zero or more pools of IP addresses that can be used for assigning floating IP addresses. Floating IP addresses in this context is specific to the Neutron floating IP address resource.

Resource	Description
NAT Pool	A pool of IP addresses that is used to assign floating IP addresses for virtual machines.

In this example the **1.105.2.128/25** range can be anything as long it is routable. Also it does not have to correlate to a CIDR in a configuration.

The following command creates a NAT pool from the subnet that is used for the external subnet:

```
# gbp nat-pool-create --ip-version 4 --ip-pool 1.105.2.128/25
--external-segment Datacenter-Out my-nat-pool
```

The following command creates a Network Service Policy (NSP) that uses the NAT pool that you created by the previous command:

```
# gbp network-service-policy-create --network-service-params
type=ip_pool,name=nat_fip,value=nat_pool my-nat-pool-nsp
```

The following command associates the NSP that you created by the previous command with the web policy target group:

```
# gbp group-update web --network-service-policy my-nat-pool-nsp
```

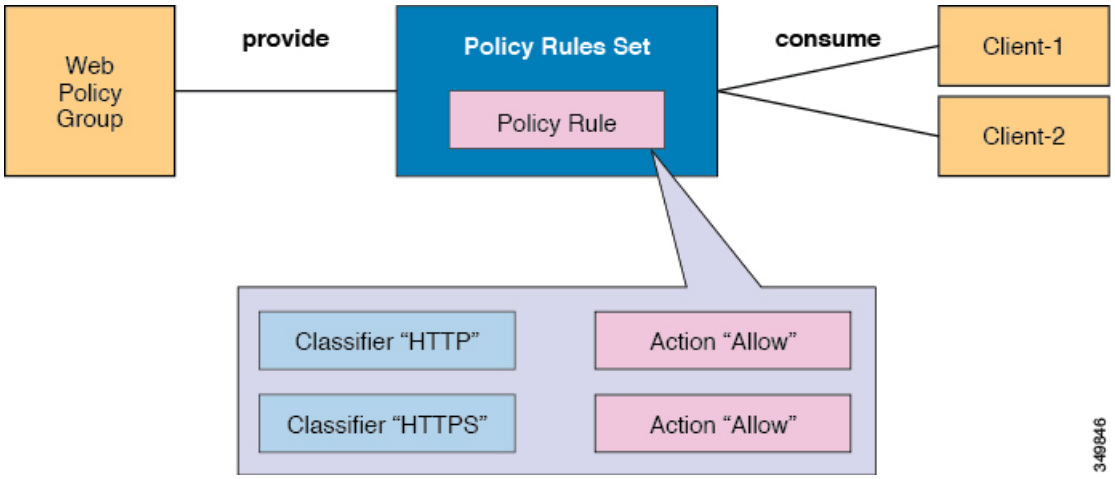
At this point, every new policy target that gets created in the web policy target group automatically gets a floating IP address assignment from the NAT pool.

The NAT pool can be any address range as long as it does not overlap with the Application Policy Infrastructure Controller (APIC) external network configuration as specified in the configuration file by the `cidr_exposed` and `host_pool_cidr` options.

The above workflow should not be used if the user wants to assign floating IP addresses explicitly only to specific members of a policy target group. In such cases, do not create an NSP, but instead follow the explicit workflow for associating a floating IP address to a Neutron port. Choose the floating IP address from the subnet that corresponds to the NAT pool that you created.

Example of Creating a Multi-Tier Application Using the Group-Based Policy CLI

The following example creates a simple policy using Group-Based Policy (GBP). This policy creates two groups and a policy rule set between them.



Procedure

Step 1 Set the rules and rule set, which describes a policy for a set of Web servers.

The rules contain classifiers designed to match a portion of the traffic and actions for dealing with that traffic. Common actions include actions to allow or redirect traffic to a network service.

a) Create the `allow` action:

```
# gbp policy-action-create allow --action-type allow
```

b) Create the HTTP rule:

```
# gbp policy-classifier-create web-traffic --protocol tcp --port-range 80 \
--direction in
# gbp policy-rule-create web-policy-rule --classifier web-traffic --actions allow
```

c) Create the HTTPS rule:

```
# gbp policy-classifier-create secure-web-traffic --protocol tcp --port-range 443 \
--direction in
# gbp policy-rule-create secure-web-policy-rule --classifier secure-web-traffic \
--actions allow
```

d) Create the Web rule set:

```
# gbp policy-rule-set-create web-ruleset --policy-rules "web-policy-rule
secure-web-policy-rule"
```

Step 2 Create the groups and associate the rule sets.

Rule sets describe a bidirectional set of rules. However, the API is designed to allow a group to "provide" a rule set that describes its behavior, and other groups to "consume" that rule set to connect to it. The model intends for groups to provide rule sets that describe their behavior, which other groups can then choose to access.

a) Create the groups:

```
# gbp group-create web
# gbp group-create client-1
# gbp group-create client-2
```

b) Associate the rule sets:

```
# gbp group-update client-1 --consumed-policy-rule-sets "web-ruleset=scope"
# gbp group-update client-2 --consumed-policy-rule-sets "web-ruleset=scope"
# gbp group-update web --provided-policy-rule-sets "web-ruleset=scope"
```

Step 3 Create a member within each of the previously-created groups.

Each member inherits all of the properties of the group to specify its connectivity and security requirements.

a) Create a policy target in the Web group, and extract the Neutron port that is associated with that policy target:

```
# export WEB_PORT=$(gbp policy-target-create web-pt-1 --policy-target-group web \
| awk "/port_id/ {print \$4}")
```

b) Create a Web group member (a virtual machine instance) using the Neutron port that you extracted earlier:

```
# nova boot --flavor m1.tiny --image image_name --nic port-id=$WEB_PORT web-vm-1
```

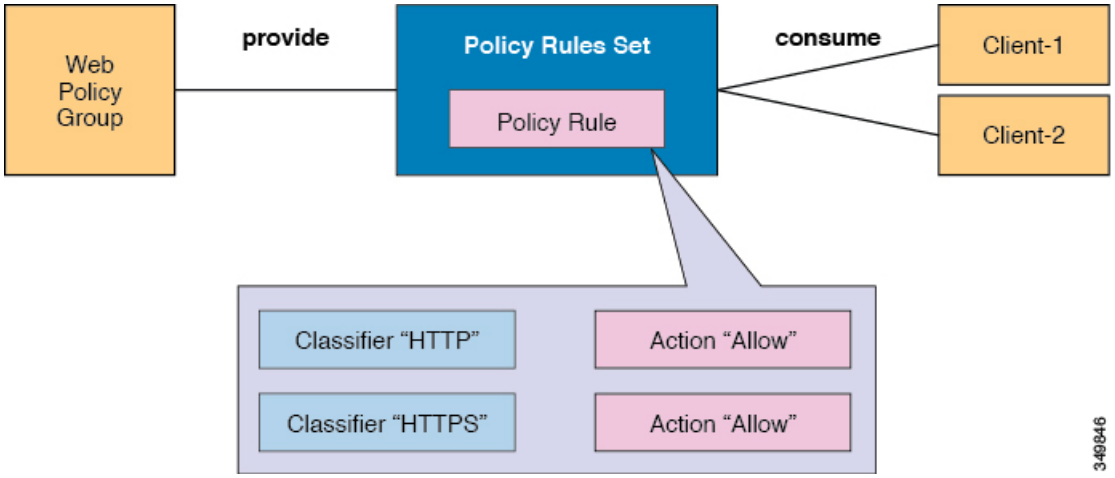
c) Create members in the `client-1` and `client-2` groups:

```
# export CLIENT1_PORT=$(gbp policy-target-create client-pt-1 \
--policy-target-groupclient-1 | awk "/port_id/ {print \$4}") nova boot \
--flavor m1.tiny --image image_name --nic port-id=$CLIENT1_PORT \
client-vm-1

# export CLIENT2_PORT=$(gbp policy-target-create client-pt-2 \
--policy-target-group client-2 | awk "/port_id/ {print \$4}") nova boot \
--flavor m1.tiny --image image_name --nic port-id=$CLIENT2_PORT \
client-vm-2
```

Example of Creating a Multi-Tier Application Using the Group-Based Policy GUI

The following example creates a simple policy for a multi-tier application using Group-Based Policy (GBP). This policy creates two groups and a policy rule set between them.



Procedure

Step 1 Log into the OpenStack platform GUI as an administrator.

Step 2 On the menu bar, choose **Project** > **Policy** > **Application Policy** > **Policy Rule Set**.

Step 3 In the **Policy Rule Set** pane, click **Create Policy Rule Set**.

Step 4 In the **Create Policy Rule Set** dialog box, perform the following actions:

a) In the **Name** field, enter the name of the new policy rule set (web-ruleset).
b) In the **Description** field, enter the description of the new policy rule set (web-ruleset).
c) Click **Next**.
d) In the **Policy Rules** field, click the **+** icon.

Step 5 In the **Create Policy-Rule** dialog box, perform the following actions to create a rule for HTTP:

a) In the **Name** field, enter the name of the new policy-rule (http-rule).
b) In the **Description** field, enter the description of the new policy-rule (http-rule).
c) Click **Next**.
d) In the **Policy Classifier** field, click the **+** icon.

Step 6 In the **Create Policy Classifier** dialog box, perform the following actions:

a) In the **Name** field, enter the name of the new policy classifier (http-classifier).
b) In the **Port/Range(min:max)** field, enter the port range (80).
c) In the **Direction** drop-down list, choose the direction (BI).
d) Click **Create Policy Classifier**.
e) Click **Next**.
f) In the **Policy Action** field, choose the action for your policy-rule (allow).
g) Click **Create**.

Step 7 In the **Create Policy Rule Set** dialog box, perform the following actions for HTTPS:

a) In the **Policy Rules** field, click the **+** icon.
b) In the **Name** field, enter the name of the new policy-rule (https-rule).
c) In the **Description** field, enter the description of the new policy-rule (https-rule).
d) Click **Next**.
e) In the **Policy Classifier** field, click the **+** icon.

Step 8 In the **Create Policy Classifier** dialog box, perform the following actions:

a) In the **Name** field, enter the name of the new policy classifier (https-classifier).
b) In the **Port/Range(min:max)** field, enter the port range (443).
c) In the **Direction** drop-down list, choose the direction (BI).
d) Click **Create Policy Classifier**.

Step 9 In the **Create Policy-Rule** dialog box, perform the following actions:

a) Click **Next**.
b) In the **Policy Action** field, choose the action for your policy-rule (allow).
c) Click **Create**.
d) In the **Policy Rules** field, choose the policy rules for your policy rule set (http-rule and https-rule).
e) Click **Create**.

Step 10 On the menu bar, choose **Project** > **Policy** > **Application Policy** > **Policy Rule Set**.

Step 11 In the **Policy Rule Set** pane, you can verify your rule set exists including the policy rules.

Step 12 Click **Policy Rules**.

Step 13 In the **Policy Rules** pane, you can verify your policy rules exists.

Step 14 Click **Policy Classifier**.

Step 15 In the **Policy Classifier** pane, you can verify your policy classifier exists.

Step 16 Click **Policy Actions**.

Step 17 In the **Policy Actions** pane, you can verify the previous created policy action exists.

Step 18 Create the policy target group. On the menu bar, choose **Project** > **Policy** > **Groups**.

Step 19 Click **Create Group**.

Step 20 In the **Create Group** dialog box, perform the following actions:

a) In the **Name** field, enter the name of the new group (web-group).
b) In the **Description** field, enter the description of the new group (web-group).
c) Click **Next**.
d) In the **Provided Policy Rule Set** field, choose the policy rule set for the group (web-ruleset).
e) Click **Next**.
f) In the **Network Policy** drop-down list, choose the network policy for the group (Default). Choosing **Default** creates a Layer 2 policy automatically that has the same name as the group.
g) Click **Create**.

Step 21 Create a second group for clients. Click **Create Group**.

Step 22 In the **Create Group** dialog box, perform the following actions:

a) In the **Name** field, enter the name of the new group for clients (client-group).
b) In the **Description** field, enter the description of the new group for clients (client-group).
c) Click **Next**.
d) In the **Consumed Policy Rule Set** field, choose the policy rule set to be consumed for the group (web-ruleset).
e) Click **Next**.
f) In the **Network Policy** field, choose the network policy for the group (Default). Choosing **Default** puts the group in the default Layer 2 policy that was implicitly created in Step 20, on page 12.
g) Click **Create**.

Step 23 On the menu bar, choose **Project** > **Policy** > **Network and Services' Policy**.

Step 24 In the **L3 Policy** pane, you can verify that the default Layer 3 policy was created.

Step 25 Click the network policy **default** to verify the Layer 2 policies were created.

Step 26 Create virtual machines. On the menu bar, choose **Project** > **Policy** > **Groups**.

Step 27 In the **Groups** pane, click on a group (web-group).

Step 28 In the **Members** pane, click **Create Member**.

Step 29 In the **Create Member** dialog box, perform the following actions:

a) In the **Instance Name** field, enter the instance name (web1).
b) In the **Instance Boot Source** drop-down list, choose the instance boot source (Boot from image).
c) In the **Image Name** drop-down list, choose the image (cirros-web).
d) Click **Launch**.
e) In the **Members** pane, you can verify the newly created group (web-group).

Step 30 Create the client virtual machine. On the menu bar, choose **Project** > **Policy** > **Groups**.

Step 31 In the **Groups** pane, click on a group (client-group).

Step 32 In the **Members** pane, click **Create Member**.

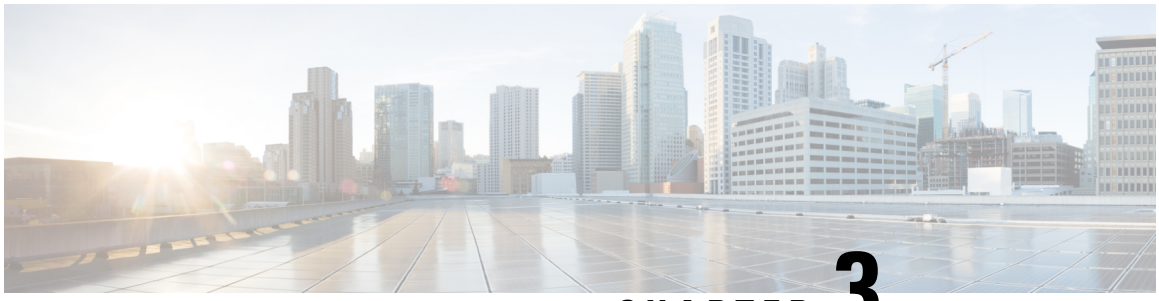
Step 33 In the **Create Member** dialog box, choose **Details**, and perform the following actions:

a) In the **Instance Name** field, enter the instance name (client1).
b) In the **Instance Boot Source** drop-down list, choose the instance boot source (Boot from image).
c) In the **Image Name** drop-down list, choose the image (cirros-web).
d) Click **Launch**.
e) In the **Members** pane, you can verify the newly created group (client-group).

Step 34 To view all instances, on the menu bar, choose **Admin** > **System** > **Instances**.

Group-Based Policy Deployment

For information on deploying Group-Based Policy (GBP), see: https://wiki.openstack.org/wiki/GroupBasedPolicy#Try_Group-based_Policy



CHAPTER 3

Neutron SVI Integration

This chapter contains the following sections:

Neutron SVI Integration Overview

A Switched Virtual Interface (SVI) is a virtual local area network (VLAN) of switch ports that have a single interface to a routing or bridging system. In the context of a Layer 3 Out (L3Out) configuration, you configure an (SVI) to provide connectivity between the Cisco Application Centric Infrastructure (ACI) leaf switch and a router.

By default, when you configure a single L3Out with SVI interfaces, the VLAN encapsulation spans multiple nodes within the fabric. The spanning occurs because the Cisco ACI fabric configures the same bridge domain (VXLAN VNI) across all the nodes in the fabric where the L3Out SVI is deployed. The bridge domain configuration occurs as long as all SVI interfaces use the same external encapsulation (SVI). However, when different L3Outs are deployed, the Cisco ACI fabric uses different bridge domains even if they use the same external encapsulation (SVI).

Note Beginning with Cisco Application Policy Infrastructure Controller (APIC) Release 5.1(1), Cisco APIC supports a dual stack on SVI networks; that is, it now supports IPv6 and IPv4 connections. This feature enables Border Gateway Protocol (BGP) peering over L3Outs for IPv6 routes and IPv4 routes if you enable them. See the section Configure SVI Integration with Dual-Stack IP Addressing, on page 18.

Neutron SVI

Beginning with Cisco APIC Release 5.1.(1), you can enable the Neutron SVI feature for VMs on OpenStack compute nodes using the OpFlex agent as well as the community OVS agent. This feature is only available in the unified mode with the AIM based plug-in.

When an OpenStack cluster is integrated with Cisco ACI through the Cisco ACI Modular Layer 2 (ML2) plug-in, you can create a floating L3Out that dynamically instantiates SVIs on Cisco ACI border leaf switches to peer with the OpenStack workload. In the Cisco ACI naming convention, this function is called *Neutron*

SVI. The Neutron SVI feature allows you to configure in OpenStack Neutron networks that automatically create Cisco ACI L3Out, which potentially can enable BGP peering.

The OpenStack administrator can bind OpenStack virtual network functions (VNFs) directly to those Neutron networks. The Cisco ACI ML2 plug-in for OpenStack dynamically creates and deletes the SVI configuration on the L3Out to peer BGP with the VNFs that are created or destroyed. You can create an SVI network without specifying the L3Out. In that case, the plug-in automatically creates it and establishes the mapping.

Reasons for Configuring SVIs

SVIs are configured for a VLAN for the following reasons:

- To allow BGP peering between virtual machines (VMs) peering with the (L3Out)
- To use the upstream OpenStack API to control L3Out node profiles
- To enable the OpenStack API to create an L3Out configuration on Cisco ACI.

SVI advantages include:

- Configuration of the dynamic routing protocol between fabric switch and VNFs
- Support for dynamic and distributed VNFs even across multiple Cisco ACI pods
- Equal-cost multipath (ECMP) traffic distribution among VNFs
- Optimal performance with VNFs
- Distributed route peering between the switches and OpenStack VNFs

  The Cisco ACI plug-in for OpenStack enables the route peering based on the creation or destruction of VNFs. The Neutron SVI feature dynamically and automatically creates and destroys the SVI on the underlay. The feature also enables line rate routing capabilities and up to 64-way ECMP to the VNFs.

  Neutron SVI supports up to six pairs of switches under same L3out. Supports VNFs across distributed sites (multipod) and bonding with VPC to fabric with bidirectional forwarding detection (BFD) for fast VM failure detection.

Configuring SVI

This section describes how to configure the Switched Virtual Interface (SVI).

Procedure

Create neutron network with "--apic:svi True":

Example:

```
#######
#creates the LB SVI network and its subnet which will be used for BGP peering between
#ACI leaf and LB --no-dhcp is required initially not to #assign a random IP to the SVI

neutron net-create LBSVI --provider:network_type vlan --provider:physical_network physnet1
 \
--apic:svi True --apic:bgp_enable True --apic:bgp_asn 2010
openstack subnet create --ip-version 4 --subnet-range 172.168.0.0/24 --gateway 172.168.0.1
```

```
 \
--network LBSVI LBSUBNET --no-dhcp

#defines the static leaf IP address for the SVI (this is optional but nice to have so the
LB
#knows the neighbor to peer with)
openstack port create apic-svi-port:node-101 --network LBSVI --device-owner apic:svi \
--fixed-ip subnet=LBSUBNET,ip-address=172.168.0.11
openstack port create apic-svi-port:node-102 --network LBSVI --device-owner apic:svi \
--fixed-ip subnet=LBSUBNET,ip-address=172.168.0.12

#now that static ports are set dhcp can be enabled
openstack subnet set LBSUBNET --dhcp

#create 2 LB VMs with static IP 172.168.0.21 and 172.168.0.22
openstack port create LB1PORT --network LBSVI --fixed-ip
subnet=LBSUBNET,ip-address=172.168.0.21 \
--fixed-ip subnet=LBSUBNET6,ip-address=2001:db8::11
openstack port create LB2PORT --network LBSVI --fixed-ip
subnet=LBSUBNET,ip-address=172.168.0.22 \
--fixed-ip subnet=LBSUBNET6,ip-address=2001:db8::12

LB1=$(openstack port list | awk '/LB1/ {print $2}')
LB2=$(openstack port list | awk '/LB2/ {print $2}')

nova boot --flavor m1.tiny --image LB1 --nic port-id=$LB1 vLB1
nova boot --flavor m1.tiny --image LB2 --nic port-id=$LB2 vLB2
```

Dual Stack with SVI Integration

OpenStack supports dual-stack IP addressing—the ability configure IPv4 and IPv6 traffic at the same time—on Switched Virtual Interface (SVI) networks. Using dual-stack IP addressing provides the advantages of IPv6—increased network efficiency and larger address space—while supporting hosts or applications that do not support IPv6.

Because of dual-stack IP addressing, the Cisco Application Centric Infrastructure (ACI) ML2 plug-in adds support for Border Gateway Protocol (BGP) peering for IPv6 connections and IPv4 connections if you enable BGP. OpenStack previously supported BGP peering for IPv4 connections.

Configuring SVI integration with dual-stack IP addressing is similar to configuring SVI integration without dual-stack. With IPv4 only, you create an SVI-type network, define a static IP address for the leaf switch, and enable DHCP. However, for dual-stack, you add both IPv4 and IPv6 subnets to the SVI-type network. You can have only one IPv4 subnet and one IPv6 subnet on the SVI network.

Adding subnets creates a bound DHCP port on Cisco Application Policy Infrastructure Controller (APIC). Bringing up a virtual machine (VM) also creates a bound port on Cisco APIC. The presence of a bound port, regardless of origin, prompts the OpenStack plug-in to create the required interface profiles. The profiles enable traffic to flow through the leaf switch.

To use BGP peering, you must enable BGP on the Neutron network creation command. For example:

```
neutron net-create LBSVI --provider:network_type vlan \
--provider:physical_network physnet1 --apic:svi True --apic:bgp_enable True \
--apic:bgp_asn 2010
```

The option **--apic:bgp_asn 2010** prepares the Layer 3 outside connection (L3Out) definition for BGP peering to the OpenStack workload. BGP configuration on the L3Out is dynamically configured only when the first OpenStack virtual machine (VM) is attached to this Neutron network. Until VMs exist, the SVI and related BGP configuration does not exist in Cisco Application Centric Infrastructure (ACI).

Note You must manually define the BGP configuration on the OpenStack workload because it is outside of the ACI ML2 plug-in control.

Prerequisites for Configuring SVI Integration with Dual Stack IP Addressing

This section lists the tasks what you must fulfill before you configure dual-stack Switched Virtual Interface (SVI) integration and Border Gateway Protocol (BGP) peering.

1. Create a tenant.
2. If you plan to specify IP addresses for the IPv4 or IPv6 connections, determine and reserve the addresses.

   Alternatively, you can let the OpenStack plug-in assign IP addresses automatically.
3. If you plan to specify a Neutron port—referred to as an SVI port in OpenStack—create one.

   Alternatively, you can let the OpenStack plug-in create one automatically.
4. If you want to use BGP peering, configure BGP routing on the virtual machine (VM) that you plan to bring up in OpenStack.
5. Create at least one virtual machine (VM).

Configure SVI Integration with Dual-Stack IP Addressing

To configure Switched Virtual Interface (SVI) integration with dual-stack IP addressing, you enter a series of OpenStack commands and verify the configuration in Cisco Application Policy Infrastructure Controller (APIC).

Note OpenStack supports only one subnet from each address type. That is, you can configure only one IPv4 subnet and only one IPv6 subnet. If you configure more than one IPv4 subnet or more than one IPv6 subnet, the OpenStack plug-in raises an error.

Before you begin

Fulfill the conditions and tasks in the section Prerequisites for Configuring SVI Integration with Dual Stack IP Addressing, on page 18.

Procedure

Step 1 Create an SVI-type network.

Example:

```
neutron net-create LBSVI --provider:network_type vlan --apic:svi True \
--apic:bgp_enable True --apic:bgp_asn 2010 \
--apic:distinguished_names type=dict
ExternalNetwork=uni/tn-common/out-Access-Out/instP-data_ext_pol
```

The command also enables Border Gateway Protocol (BGP) on the network. If you configure an endpoint on the network, the OpenStack plug-in creates a BGP peer connectivity profile on Cisco APIC.

The option **--apic:distinguished_names type-dict ...._pol** is optional. You need it only if you want to use an existing Layer 3 outside (L3Out) connection already defined on Cisco Application Centric Infrastructure (ACI). If you do not give this parameter, a new L3Out is created on the Cisco ACI fabric from the OpenStack plug-in.

Step 2 Add an IPv4 subnet to the network.

Example:

```
openstack subnet create --ip-version 4 --subnet-range 172.168.0.0/24  \
--gateway 172.168.0.1 --network LBSVI LBSUBNETP-data_ext_pol68.0.11
```

The OpenStack plug-in creates a DHCP port on Cisco APIC. The DHCP port is a bound port, and its creation triggers the static path.

Step 3 Create a Neutron port, which the interface profile requires.

You can skip the step and allow the OpenStack plug-in to create the port automatically.

In OpenStack, the Neutron port is referred to as the SVI port.

Example:

```
openstack port create apic-svi-port:node-103 --network LBSVI --device-owner apic:svi \
--fixed-ip subnet=LBSUBNET,ip-address=172.168.0.11
```

Creation of the DHCP and SVI ports automatically triggers the creation of an interface profile on Cisco APIC, which enables traffic to flow through the leaf switch.

Step 4 Add an IPv6 subnet.

Example:

```
openstack subnet create --ip-version 6 --subnet-range 2001:db8::/64 --gateway 2001:db8::1
\
--ipv6-ra-mode slaac --ipv6-address-mode slaac --network LBSVI LBSUBNET6
```

Neutron updates the DHCP port with the new subnet information, and the OpenStack plug-in automatically creates an interface profile for the new subnet. You created Neutron (SVI) ports with IPv4 and IPv6 addresses in the section Configuring SVI, on page 16.

What to do next

Verify the SVI integration in Cisco APIC:

Go to **Tenants** > ***tenant*** > **Networking** > **L3Outs** > **Access-Out** > **Logical Node Profiles** > **IfProfile** or **IfProfile6**, and then click the **SVI** tab. The central work pane displays the path, IP address, and other information about the interface profile.

Configure BGP

After you configure a Switched Virtual Interface (SVI) with Border Gateway Protocol (BGP) enabled, you can configure BGP for IPv4. Beginning with Cisco Application Policy Infrastructure Controller (APIC) Release 5.1.1, you can configure BGP for IPv6 routes and IPv4 routes.

Before you begin

You must have enabled BGP when you configured an SVI. See Configure SVI Integration with Dual-Stack IP Addressing, on page 18.

Procedure

Step 1 Configure BGP on OpenStack virtual machines (VMs) on the SVI network with the appropriate export routes.

The BGP configuration is separate from OpenStack or Cisco Application Centric Infrastructure (ACI) and depends on the VM type.

Step 2 Verify the BGP configuration.

a) Log in to Cisco APIC.
b) Go to **Tenants** > ***tenant*** > **L3Outs** > **Access-Out** > **Logical Node Profiles** > **NodeProfile** > **Configured Nodes** > **topology/*pod*/*node*** > **BGP for VRF-common:** > **Neighbors**.
c) Open the **Neighbors** folder.
d) The folder displays the IP address for the IPv4 route and, if you configured it, the IP address for the IPv6 route.

Troubleshooting SVI Networks

This section describes how to troubleshoot SVI networks.

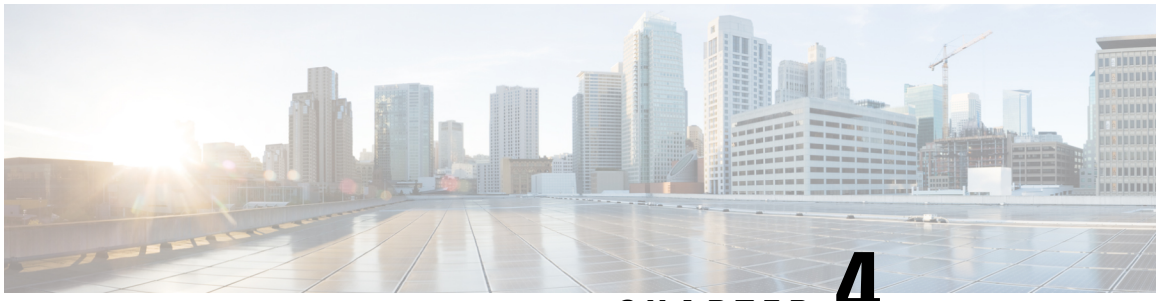
- Make sure that the l3 domain DN is configured properly. The l3 domain DN pointing to a working external routed domain in APIC in the neutron config file and neutron-server has been restarted on all the controller nodes.
- Make sure that there are no faults in either the pre-existing l3-outs or the auto l3-outs created by our mechanism driver.
- Make sure that SVI interface is being created properly with the right path and the VLAN ID of the SVI network when DHCP or VM endpoints are getting created.
    - In a VPC setup, the primary IP of site A and site B will be allocated from the SVI subnet by our mechanism driver and consistent per VPC pair across the SVI interfaces. The secondary IP will always be the Gateway IP of the SVI subnet.
- Make sure that nodes are getting created properly under node profile in the l3-out.
    - When SVI interface is being created, our mechanism_driver will create the corresponding nodes also. The nodes info is in the SVI path itself. In a VPC setup, each SVI interface will have 2 nodes in its path.

- `Bgp_asn` parameter that is specified at the time of network create or update should be the same as the one used for peering by the guest machine acting as a bgp peer and it should not match the AS number that is used by ACI for internal fabric BGP peering. Also ensure that provider and consumer BGP peers have different AS numbers in order to redistribute routes from one peer to the other.

Note Directly connected subnets are never redistributed as is the common use case with eBGP. If needed, these subnets can be exported with an explicit permit route-map outside of openstack, by posting to APIC.

- Minimal config to establish BGP session and import/export prefixes that are learned by peering to other peers is exposed with openstack integration, for all advanced route-map/community/password configuration, use the APIC API directly.
- Use the **aimctl manager** command to debug if you think things should be created under that l3-out but you do not see it in APIC or vice versa.
  - Just type **aimctl manager | grep out** and **aimctl manager | grep external** then it will list all the l3-out and external-network related CLI commands available to use.
  - If `sync_status` shows some failure, check the `/var/log/aim-aid.log` file for more details.

Note If you use OSP16 or later, you must execute the aimctl commands from inside the ciscoaci_aim container. The container can be entered from a controller node by running the following command: **`docker exec -itu root ciscoaci_aim bash.`** To exit the container, type **`exit`**.



CHAPTER 4

Neutron SFC Integration

This chapter contains the following sections:

Neutron SFC Integration Overview

You can check with your OpenStack provider for CLI and GUI support.

The reasons why you should use Neutron SFC Integration:

- Makes multinode PBR easier
- Use upstream Openstack API to deploy Service Graph with Multinode PBR
- No ACI manual configuration
- Framework for creating service chains using the Neutron API

Configuring Neutron SFC Using the CLI

This section describes how to configure Neutron service function chaining (SFC) using the CLI which have not been tested or supported and are only for informational propose. The CLI reference is provided in the link below:

https://docs.openstack.org/neutron/queens/admin/config-sfc.html

Here are the exception:

1. Cisco only supports configuring Neutron SFC using the REST API.
2. Out of the traffic classifiers from the upstream project our driver only supports these:
   - `source_ip_prefix`- Source IP address or prefix
   - `destination_ip_prefix` - Destination IP address or prefix

Before you begin

Check with your OpenStack provider for CLI and GUI support.

Procedure

Step 1 Create Left and Right Networks (BD):

Example:

```
neutron net-create SRC-NET

openstack subnet create --ip-version 4 --gateway 1.1.0.1 --network SRC-NET_ID \
--subnet-range 1.1.0.0/24 --host-route destination=10.0.0.0/16,gateway=1.1.0.1 ''

neutron net-create DST-NET

openstack subnet create --ip-version 4 --gateway 2.2.0.1 --network DST-NET_ID \
--subnet-range 2.2.0.0/24 --host-route destination=0.0.0.0/0,gateway=2.2.0.1 ''
```

a) Create Flow Classifier:

Example:

```
neutron flow-classifier-create --destination-ip-prefix 0.0.0.0/0 --source-ip-prefix \
10.0.1.0/24 --l7-parameters logical_source_network=\
SRC-NET_ID,logical_destination_network=DST-NET_ID CLASSIFIER1
```

Step 2 Create Src and Dest neutron Ports:

Example:

```
openstack port create SERVICE1-INGRESS --network SRC-NET_ID --no-security-group \
--disable-port-security --fixed-ip subnet=SRC-SUBNET_ID,ip-address=1.1.0.11

openstack port create SERVICE1-EGRESS --network DST-NET_ID --no-security-group \
--disable-port-security --fixed-ip subnet=DST-SUBNET_ID,ip-address=2.2.0.11
```

Step 3 Create Port Pair:

Example:

```
neutron port-pair-create --ingress SERVICE1-INGRESS-PORT_ID --egress \
SERVICE1-EGRESS-PORT_ID PORTPAIR1
```

Step 4 Create Port Pair Group:

Example:

```
neutron port-pair-group-create --port-pair PORTPAIR1_ID CLUSTER1
```

Step 5 Create Service Chain:

Example:

```
neutron -port-chain-create --flow-classifier CLASSIFIER1_ID --port-pair-group CLUSER1_ID \
SERVICE-CHAIN1
```

Note Once the service chain is established, the egress and ingress interfaces on the service VM (VNF) can only be used for redirected traffic. For example, the DHCP on these interfaces will not be functional. It is recommended to have a separate management interface for managing the VNF configuration including IP configuration of the egress and ingress interface.

Step 6 Create service VM:

Example:

```
nova boot --flavor medium --image ServiceImage1 --nic \
port-id=SERVICE1-INGRESS-PORT_ID --nic port-id=SERVICE1-INGRESS-_PORT_ID SERVICE-VM-1
```

Step 7 Add bumps on a wire:

a) Create more left and right networks (BD):

Example:

```
neutron net-create SRC-NET2

openstack subnet create --ip-version 4 --gateway 1.1.0.1 --network SRC-NET2_ID \
--subnet-range 1.1.0.0/24 --host-route destination=10.0.0.0/16,gateway=1.1.0.1 ''

neutron net-create DST-NET2

openstack subnet create --ip-version 4 --gateway 2.2.0.1 --network DST-NET2_ID \
--subnet-range 2.2.0.0/24 --host-route destination=0.0.0.0/0,gateway=2.2.0.1 ''
```

b) Create Src and Dest neutron Ports for service 2:

Example:

```
openstack port create SERVICE2-INGRESS --network SRC-NET2_ID --no-security-group \
--disable-port-security --fixed-ip subnet=SRC-SUBNET2_ID,ip-address=3.3.0.11

openstack port create SERVICE2-EGRESS --network DST-NET2_ID --no-security-group \
--disable-port-security --fixed-ip subnet=DST-SUBNET2_ID,ip-address=4.4.0.11
```

Step 8 To add more bumps on a wire:

a) Create Port Pair for service 2:

Example:

```
neutron port-pair-create --ingress SERVICE2-INGRESS_PORT_ID --egress \
SERVICE2-EGRESS_PORT_ID PORTPAIR2
```

b) Create Port Pair Group for service 2:

Example:

```
neutron port-pair-group-create –port-pair PORTPAIR1_ID CLUSTER2
```

c) Update Service Chain (add new Port Pair Group):

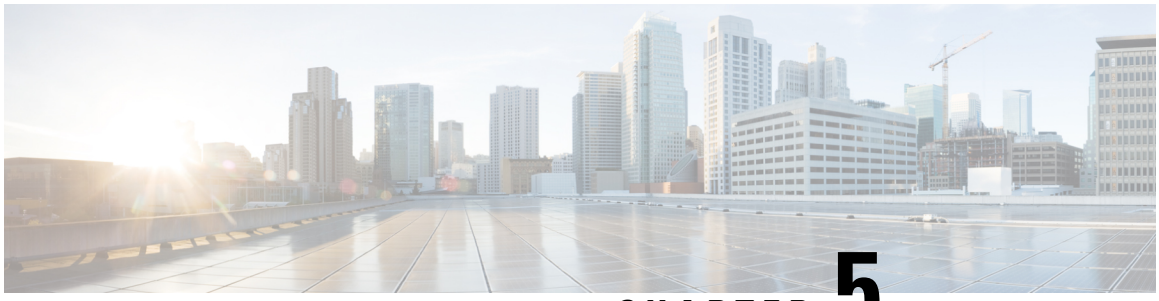
Example:

```
neutron port-chain-update SERVICE-CHAIN1_ID --flow-classifier CLASSIFIER1_ID \
--port-pair-group CLUSTEER1 --port-pair-group CLUSTER2
```

d) Create service2 VM:

Example:

```
nova boot --flavor medium --image ServiceImage1 --nic \
port-id=SERVICE2-INGRESS-PORT_ID --nic port-id=SERVICE2-EGRESS-PORT_ID SERVICE_VM-2
```



CHAPTER 5

Day 2 Operations

This chapter contains the following sections:

Data Plane Verification

Data Plane Verification Overview

This chapter describes how to verify the Cisco ACI plugin for OpenStack has been installed correctly, how to test basic connectivity of an OpenStack instance with its default gateway and with the externally created network.

Prerequisite

Before you get started, make sure that you have met the following prerequisites:

- Make sure the commands are executed sourcing the keystone file for the project where you want to create the network constructs and instance.
- Make sure one external network called external-network-shared has already been created as a shared resource for any OpenStack project.
- Make sure Nova already has a pre-defined flavor in order to create instances.
- Make sure Glance already has a pre-defined with an image to boot instances.

Creating a Neutron Network

This section describes how to create a Neutron network.

Procedure

Step 1 Create a Neutron network, enter the following command:

```
$ openstack network create test_net
```

Sample output:

```
+---------------------------+-------------------------------------------------+
|Field                      |Value                                            |
|---------------------------+-------------------------------------------------
| admin_state_up            | UP                                              |
| availability_zone_hints   |                                                 |
| availability_zone         |                                                 |
| created_at                | 2018-07-24T20:32:27z                            |
| description               |                                                 |
| dns_domain                | None                                            |
| id                        | 016b9885-c8ac-4a2d-be7e-e5203c945ba4            |
| ipv4_address_scope        | None                                            |
| ipv6_address_scope        | None                                            |
| is_default                | None                                            |
| mtu                       | 1500                                            |
| name                      | test_net                                        |
| port_security_enabled     | True                                            |
| project_id                | 7d0be879a12c47ae9c0a26d3fd4407d1                |
| provider:network_type     | opflex                                          |
| provider:physical_type    | physnet1                                        |
| provider:segmentation_id  | None                                            |
| qos_policy_id             | None                                            |
| revision_number           | 3                                               |
| router:external           | Internal                                        |
| segments                  | None                                            |
| shared                    | False                                           |
| status                    | ACTIVE                                          |
| subnets                   |                                                 |
| updated_at                | 2018-07-24T20:32:27Z                            |
+---------------------------+-------------------------------------------------+
```

Similarly, ACI fabric, it should be possible to verify that in the corresponding tenant the EPG and BD with the Neutron network name have been defined like in

Step 2 Verify the EPG and BD with the Neutron network name have been defined.

a) In the APIC GUI, on the menu bar, choose **Tenants** > ***tenant_name*** > **Application Profiles** > **OpenStack** > **Application EPGs** > ***EPG_name***. Check if the EPG has been defined.

b) In the APIC GUI, on the menu bar, choose **Tenants** > ***tenant_name*** > **Networking** > **Bridge Domains** > ***BD_name***. Check if the BD has been defined.

Creating a Neutron Subnet

This section describes how to create a Neutron Subnet.

Procedure

Create a Neutron subnet and bound to the network previously created, enter the following CLI command:

Example:

```
openstack subnet create --network test_net --gateway 192.168.1.254 \
--subnet-range 192.168.1.0/24 subnet01
```

Sample output:

```
+---------------------------+------------------------------------------------+
|Field                      |Value                                           |
|---------------------------+------------------------------------------------
| allocation_pools          | 192.168.1.1-192.168.1.253                      |
| cidr                      | 192.168.1.0/24                                 |
| created_at                | 2018-07-24T20:37:03Z                           |
| description               |                                                |
| dns_nameservers           |                                                |
| enable_dhcp               | True                                           |
| gateway_ip                | 192.168.1.254                                  |
| host_routes               |                                                |
| id                        | d3341f6d-5fbe-476e-a0b7-d0e1b546eba4           |
| ip_version                | 4                                              |
| ipv6_address_mode         | None                                           |
| ipv6_ra_mode              | None                                           |
| name                      | subnet01                                       |
| network_id                | 016b9885-c8ac-4a2d-be7e-e5203c945ba4           |
| project_id                | 7d0be879a12c47ae9c0a26d3fd4407d1               |
| revision_number           | 2                                              |
| segment_id                | None                                           |
| service_types             |                                                |
| subnetpool_id             | None                                           |
| updated_at                | 2018-07-24T20:37:03Z                           |
+---------------------------+------------------------------------------------+
```

This command will not result in any change on ACI. The subnet is not yet attached to any router and this is not enabled for L3 routing. Therefore the ACI bridge domain keeps not having any subnet associated yet.

Creating a Neutron Router

This section describes how to create a Neutron router.

Procedure

Step 1 Create a Neutron router, enter the following CLI command:

Example:

```
openstack router create router01
```

Sample output:

```
+---------------------------+------------------------------------------------+
|Field                      |Value                                           |
|---------------------------+------------------------------------------------
| admin_state_up            | UP                                             |
| availability_zone_hints   | None                                           |
| availability_zone         | None                                           |
| created_at                | 2018-07-24T20:44:11Z                           |
| description               |                                                |
| distributed               | False                                          |
| external_gateway_info     | None                                           |
| flavor_id                 | None                                           |
| ha                        | False                                          |
| id                        | 236734ab-c39e-4ad7-a9ab-c0d1fb03691a           |
| name                      | router01                                       |
| project_id                | 7d0be879a12c47ae9c0a26d3fd4407d1               |
| revision_number           | None                                           |
| routes                    | None                                           |
```

```
| status                    | ACTIVE                                                |
| updated_at                | 2018-07-24T20:41:11Z                                  |
+---------------------------+-------------------------------------------------------+
```

This command creates an ACI contract in the ACI Common tenant. The OpenStack routers are in fact rendered as 'permit IP any any' type of contracts in ACI. The contracts are always placed in the Common tenant and then applied as consumer and provider to all the EPG created Neutron networks, which subnets are bound to that router.

Step 2 Verify that the contract created in the Common ACI tenant.

In the APIC GUI, on the menu bar, choose **Tenants** > **common** > **Tenant Common** > **Contracts** > **Standards** > ***router_name***. Check if the router has been defined.

Bind the Subnet to the Router

This section describes how to bind the subnet to the router.

Procedure

Step 1 Enable routing on the neutron network created, enter the following CLI command:

Example:

```
openstack router add subnet router01 subnet01
```

As a result, on APIC a VRF called DefaultRouterVRF will be created. The BD will be bound to this VRF and also the Neutron subnet will be created as BD subnet.

Step 2 Verify the VRF called DefaultRouterVRF was created, BD was bounded to the VRF and also the Neutron subnet was created as BD subnet.

a) In the APIC GUI, on the menu bar, choose **Tenants** > **tenant_name** > **Networking** > **Bridge Domains** > ***BD_name*** > **Subnets** > ***subnet***. Check if the subnet has been defined.

b) In the APIC GUI, on the menu bar, choose **Tenants** > **tenant_name** > **VRFs** > **DefaultRoutedVRF (DefaultVRF)**. Check if the DefaultRouterVRF has been defined.

Set a Gateway to the Router

This section describes how to set a gateway to the router.

Procedure

Step 1 In order to provide external connectivity from OpenStack domain to an external router, it is necessary to set a gateway for the OpenStack router previously created. The following command assumes that an external network defined as external-net-shared exists already and can be consumed by the OpenStack project:

Example:

```
openstack router set --external-gateway external-net-shared router01
```

Step 2 Verify that the L3out was created.

In the APIC GUI, on the menu bar, choose **Tenants** > **tenant_name** > **Networking** > **External Routed Networks** > **l3out1-DefaultVRF (l3out1-DefaultVRF)**. Check if the l3out1-DefaultVRF has been defined.

Creating an Instance and Attach its NIC to the Network Previously Created

This section describes how to create an instance and attach its NIC to the network perviously created.

Procedure

Step 1 Now that network is created and configured to be routable to the external router, an OpenStack instance can be created and attached to the Neutron network to verify the connectivity. Create a Nova VM, enter the following CLI command:

Example:

```
NET1=$(openstack network list | awk '/test_net/ {print $2}')
nova boot --flavor m1.tiny --image cirros --nic net-id=$NET1 vm1
```

Step 2 Verify the VM `vm1` is visible under the EPG `test_net` Operational tab.

In the APIC GUI, on the menu bar, choose **Tenants** > ***tenant_name*** > **Application Profiles** > ***EPG_name*** > **Application EPGs** > ***EPG***. Click on the Operational tab in the pane. Check if the VM is visible. The IP address should be correctly sensed by APIC.

Verifying through ICMP that the VM is Correctly Connected to the Rest of the Infrastructure

This section describes how to verify through ICMP that the VM is correctly connected to the rest of the infrastructure.

Procedure

Verify that ICMP connectivity from the VM to its default gateway and to an external IP is reachable through the L3out, enter the following CLI commands:

Example:

```
$ ifconfig eth0
$ ping 192.168.1.254
```

Changing DNS Parameters or Static Routes

When you update the Domain Name System (DNS) parameters or static routes for a subnet, you must take extra steps to make the changes visible at a particular port or instance. We recommend that you use one of the two following sets of steps:

- Associate and then disassociate the affected port with a dummy security group.

  A dummy security group is an extra security group that you add to any security groups already configured on the port. The dummy security group does not need to contain any rules.

- Set the adminstrative state of the affected port to down, and then set the state to up.

Either set of steps triggers an update notification for the port, which allows the port to start using the changed parameters.