Cisco APIC Layer 4 to Layer 7 Services Deployment Guide, Release 3.x and earlier

First Published: 2018-05-22

Last Modified: 2021-02-10

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

THE SPECIFICATIONS AND INFORMATION REGARDING THE PRODUCTS IN THIS MANUAL ARE SUBJECT TO CHANGE WITHOUT NOTICE. ALL STATEMENTS, INFORMATION, AND RECOMMENDATIONS IN THIS MANUAL ARE BELIEVED TO BE ACCURATE BUT ARE PRESENTED WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED. USERS MUST TAKE FULL RESPONSIBILITY FOR THEIR APPLICATION OF ANY PRODUCTS.

THE SOFTWARE LICENSE AND LIMITED WARRANTY FOR THE ACCOMPANYING PRODUCT ARE SET FORTH IN THE INFORMATION PACKET THAT SHIPPED WITH THE PRODUCT AND ARE INCORPORATED HEREIN BY THIS REFERENCE. IF YOU ARE UNABLE TO LOCATE THE SOFTWARE LICENSE OR LIMITED WARRANTY, CONTACT YOUR CISCO REPRESENTATIVE FOR A COPY.

The Cisco implementation of TCP header compression is an adaptation of a program developed by the University of California, Berkeley (UCB) as part of UCB's public domain version of the UNIX operating system. All rights reserved. Copyright © 1981, Regents of the University of California.

NOTWITHSTANDING ANY OTHER WARRANTY HEREIN, ALL DOCUMENT FILES AND SOFTWARE OF THESE SUPPLIERS ARE PROVIDED “AS IS" WITH ALL FAULTS. CISCO AND THE ABOVE-NAMED SUPPLIERS DISCLAIM ALL WARRANTIES, EXPRESSED OR IMPLIED, INCLUDING, WITHOUT LIMITATION, THOSE OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT OR ARISING FROM A COURSE OF DEALING, USAGE, OR TRADE PRACTICE.

IN NO EVENT SHALL CISCO OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, CONSEQUENTIAL, OR INCIDENTAL DAMAGES, INCLUDING, WITHOUT LIMITATION, LOST PROFITS OR LOSS OR DAMAGE TO DATA ARISING OUT OF THE USE OR INABILITY TO USE THIS MANUAL, EVEN IF CISCO OR ITS SUPPLIERS HAVE BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

Any Internet Protocol (IP) addresses and phone numbers used in this document are not intended to be actual addresses and phone numbers. Any examples, command display output, network topology diagrams, and other figures included in the document are shown for illustrative purposes only. Any use of actual IP addresses or phone numbers in illustrative content is unintentional and coincidental.

All printed copies and duplicate soft copies of this document are considered uncontrolled. See the current online version for the latest version.

Cisco has more than 200 offices worldwide. Addresses and phone numbers are listed on the Cisco website at www.cisco.com/go/offices.

Cisco and the Cisco logo are trademarks or registered trademarks of Cisco and/or its affiliates in the U.S. and other countries. To view a list of Cisco trademarks, go to this URL: https://www.cisco.com/c/en/us/about/legal/trademarks.html. Third-party trademarks mentioned are the property of their respective owners. The use of the word partner does not imply a partnership relationship between Cisco and any other company. (1721R)

© 2018–2021 Cisco Systems, Inc. All rights reserved.

CONTENTS

CHAPTER 1 **New and Changed Information** 1

New and Changed Information 1

CHAPTER 2 **Overview** 3

About Deploying Application-Centric Infrastructure Layer 4 to Layer 7 Services 3

About Layer 4 to Layer 7 Service Devices 4

About Service Graph Templates 4

Configuring Layer 4 to Layer 7 Services Using the GUI 5

CHAPTER 3 **Importing a Device Package** 7

About Device Packages 7

Installing a Device Package Using the REST API 9

Importing a Device Package Using the GUI 10

CHAPTER 4 **Defining a Logical Device** 11

About Device Clusters 11

About Managed Device Clusters 12

About Unmanaged Device Clusters 12

About Concrete Devices 12

About Trunking 13

Configuring a Layer 4 to Layer 7 Services Device Using the GUI 13

Creating a Layer 4 to Layer 7 Device Using the NX-OS-Style CLI 16

Enabling Trunking on a Layer 4 to Layer 7 Virtual ASA device Using the GUI 20

Enabling Trunking on a Layer 4 to Layer 7 Virtual ASA device Using the REST APIs 20

Modifying a Device Using the GUI 21

Using an Imported Device with the REST APIs 21

Importing a Device From Another Tenant Using the NX-OS-Style CLI 22
Verifying the Import of a Device Using the GUI 22

CHAPTER 5 Configuring Connectivity to Devices 25
About In-Band Management for Devices 25
Configuring In-Band Management for Devices Using the GUI 26
Troubleshooting In-Band Management for Devices Using the GUI 27

CHAPTER 6 Selecting a Layer 4 to Layer 7 Device to Render a Graph 29
About Device Selection Policies 29
Creating a Device Selection Policy Using the GUI 29
Configuring a Device Selection Policy Using REST APIs 32
Creating a Device Selection Policy Using the REST API 32
Adding a Logical Interface in a Device Using the REST APIs 33

CHAPTER 7 Configuring a Service Graph 35
About Service Graphs 35
About Function Nodes 37
About Function Node Connectors 37
About Service Graph Connections 38
About Terminal Nodes 38
Guidelines and Limitations for Configuring Service Graph 38
About Service Graph Template Configuration Parameters 38
Configuring a Service Graph Template Using the GUI 38
Configuring a Service Graph Template Using the REST APIs 40
Applying a Service Graph Template to Endpoint Groups Using the GUI 41
Associating a Service Graph Template with a Contract Using the NX-OS-Style CLI 42
Creating a Function Profile Using the GUI 46
Using an Existing Function Profile to Create a New Function Profile Using the GUI 47

CHAPTER 8 Configuring Route Peering 49
About Route Peering 49
Open Shortest Path First Policies 50
Border Gateway Protocol Policies 54

Selecting an L3extOut Policy for a Cluster 57
Route Peering End-to-End Flow 58
Cisco Application Centric Infrastructure Fabric Serving As a Transit Routing Domain 60
Configuring Route Peering Using the GUI 61
Creating a Static VLAN Pool Using the GUI 61
Creating an External Routed Domain Using the GUI 62
Creating an External Routed Network Using the GUI 62
Creating a Router Configuration Using the GUI 64
Creating a Service Graph Association Using the GUI 64
Configuring Route Peering Using the NX-OS-Style CLI 65
Troubleshooting Route Peering 67
Verifying the Leaf Switch Route Peering Functionality Using the CLI 68

CHAPTER 9 **Configuring Policy-Based Redirect 71**
About Policy-Based Redirect 71
Guidelines and Limitations for Configuring Policy-Based Redirect 73
Configuring Policy-Based Redirect Using the GUI 79
Configuring Policy-Based Redirect Using the NX-OS-Style CLI 80
Verifying a Policy-Based Redirect Configuration Using the NX-OS-Style CLI 83
About Multi-Node Policy-Based Redirect 85
About Symmetric Policy-Based Redirect 85
Policy Based Redirect and Hashing Algorithms 86
Policy-Based Redirect Resilient Hashing 86
Enabling Resilient Hashing in L4-L7 Policy-Based Redirect 88
PBR Support for Service Nodes in Consumer and Provider Bridge Domains 88
Policy-Based Redirect and Tracking Service Nodes 88
Policy-Based Redirect and Threshold Settings for Tracking Service Nodes 89
Guidelines and Limitations for Policy-Based Redirect Tracking With Service Nodes 89
Configuring PBR and Tracking Service Nodes Using the GUI 90
Configuring a Redirect Health Group Using the GUI 91
Configuring PBR to Support Tracking Service Nodes Using the REST API 91
About Location-Aware Policy Based Redirect 92
Guidelines for Location-Aware PBR 92
Configuring Location-Aware PBR Using the GUI 93

Configuring Location-Aware PBR Using the REST API 94

Policy-Based Redirect and Service Graphs to Redirect All EPG-to-EPG Traffic Within the Same VRF Instance 94

Guidelines and Limitations for Configuring a Policy-Based Redirect Policy with a Service Graph to Redirect All EPG-to-EPG Traffic Within the Same VRF Instance 97

Configuring a Policy-Based Redirect Policy with a Service Graph to Redirect All EPG-to-EPG Traffic Within the Same VRF Instance 97

CHAPTER 10 **Configuring Direct Server Return 101**

About Direct Server Return 101

Layer 2 Direct Server Return 102

About Deploying Layer 2 Direct Server Return with Cisco Application Centric Infrastructure 103

Guidelines and Limitations for Configuring Direct Server Return 104

Supported Direct Server Return Configuration 105

Example XML POST of Direct Server Return for Static Service Deployment 105

Direct Server Return for Static Service Deployment 106

Direct Server Return for Static Service Deployment Logical Model 106

Direct Server Return for Service Graph Insertion 106

Direct Server Return Shared Layer 4 to Layer 7 Service Configuration 107

Configuring the Citrix Server Load Balancer for Direct Server Return 107

Configuring a Linux Server for Direct Server Return 107

CHAPTER 11 **Configuring the Device and Chassis Manager 109**

About Device Managers and Chassis Managers 109

Device Manager and Chassis Manager Behavior 112

Creating Devices Using the GUI 113

Creating a Chassis Using the GUI 113

Example XML for Device Managers and Chassis Managers 114

Example XML for Creating the MDevMgr Object 114

Example XML for Associating an LDevVip Object With a DevMgr Object 114

Example XML for Creating the MChassis Object 115

Example XML for Creating the Chassis Object 115

Example XML for Associating an CDev Object With a Chassis Object 115

Device and Chassis Callouts 116

Example deviceValidate Callout for a Device 116
Example deviceAudit Callout for a Device 116
Example clusterAudit Callout for a Device 116
Example serviceAudit Callout for a Device 117
Example deviceValidate Callout for a Chassis 118
Example deviceAudit Callout for a Chassis 118
Example clusterAudit Callout for a Chassis 118
Example serviceAudit Callout for a Chassis 119

CHAPTER 12 **Configuring Unmanaged Mode** 121
About the Unmanaged Mode 121
About Managed and Unmanaged Logical Devices 121
About Managed and Unmanaged Function Nodes 122
About Layer 4 to Layer 7 Services Endpoint Groups 123
Using Static Encapsulation for a Graph Connector 123
Creating a Physical Device Using the NX-OS-Style CLI 123
Creating a High Availablity Cluster Using the NX-OS-Style CLI 125
Creating a Virtual Device Using the NX-OS-Style CLI 126
Example XML for the Unmanaged Mode 127
Example XML of Creating an Unmanaged LDevVip Object 127
Example XML of Creating an Unmanaged AbsNode Object 127
Example XML of Associating a Layer 4 to Layer 7 Service Endpoint Group with a Connector 128
Example XML of Using Static Encapsulation with a Layer 4 to Layer 7 Service Endpoint Group 128
Unmanaged Mode Behavior 129

CHAPTER 13 **Configuring Copy Services** 131
About Copy Services 131
Copy Services Limitations 132
Configuring Copy Services Using the GUI 132
Creating a Copy Device Using the GUI 133
Configuring Copy Services Using the NX-OS-Style CLI 134
Configuring Copy Services Using the REST API 136

CHAPTER 14 **Configuring Layer 4 to Layer 7 Resource Pools** 139

About Layer 4 to Layer 7 Resource Pools 139
About External and Public IP Address Pools 139
About External Layer 3 Routed Domains and the Associated VLAN Pools 140
About External Routed Networks 140
Supported Managed Mode Layer 4 to Layer 7 Devices 140
About Cloud Orchestrator Mode Function Profiles 141
Creating an IP Address Pool for Layer 4 to Layer 7 Resource Pools Using the GUI 141
Creating a Dynamic VLAN Pool for Layer4 to Layer 7 Resource Pools Using the GUI 142
Creating an External Routed Domain for Layer 4 to Layer 7 Resource Pools Using the GUI 142
Preparing Layer 4 to Layer 7 Devices for Use in Layer 4 to Layer 7 Resource Pools 143
Validating the APIC Configuration of a Layer 4 to Layer 7 Device for Use in a Layer 4 to Layer 7 Resource Pool 143
Configuring the Device Management Network and Routes 144
Creating a Layer 4 to Layer 7 Resource Pool 144
Creating a Layer 4 to Layer 7 Resource Pool Using the GUI 144
Creating a Layer 4 to Layer 7 Resource Pool Using the NX-OS-Style CLI 145
Configuring a Layer 4 to Layer 7 Resource Pool Using the GUI 146
Configuring Layer 4 to Layer 7 Devices in a Resource Pool 146
Adding Layer 4 to Layer 7 Devices to a Layer 4 to Layer 7 Resouce Pool 146
Removing Layer 4 to Layer 7 Devices from a Layer 4 to Layer 7 Resource Pool 147
Configuring External IP Address Pools in a Resource Pool 147
Adding an External IP Address Pool to a Layer 4 to Layer 7 Resource Pool 147
Removing an External IP Address Pool from a Layer 4 to Layer 7 Resource Pool 148
Configuring Public IP Address Pools in a Resource Pool 148
Adding Public IP Address Pools to a Layer 4 to Layer 7 Resource Pool 148
Removing Public IP Address Pools from a Layer 4 to Layer 7 Resource Pool 149
Updating an External Routed Domain for a Layer 4 to Layer 7 Resource Pool 149
Updating External Routed Networks for a Layer 4 to Layer 7 Resource Pool 150
Configuring Cloud Orchestrator Mode Function Profiles in a Resource Pool 150
Adding a Cloud Orchestrator Mode Function Profile to a Layer 4 to Layer 7 Resource Pool 150
Removing a Cloud Orchestrator Mode Function Profile from a Layer 4 to Layer 7 Resource Pool 151

CHAPTER 15 **Configuration Parameters** 153

Configuration Parameters Inside the Device Package Specification 153
Configuration Scope of a Device Package Specification 155
Example XML of Configuration Parameters Inside the Device Package 156
Configuration Parameters Inside An Abstract Function Profile 156
Configuration Scope of an Abstract Function Profile 158
Example XML POST for an Abstract Function Profile With Configuration Parameters 159
Configuration Parameters Inside an Abstract Function Node in a Service Graph 160
Example XML POST for an Abstract Function Node With Configuration Parameters 162
Configuration Parameters Inside Various Configuration MOs 163
Example XML POST for an Application EPG With Configuration Parameters 165
Parameter Resolution 166
Looking Up an MO During Parameter Resolution 167
About Role-Based Access Control Rule Enhancements 168
Role-Based Access Control Rule Architecture 168
Role-Based Access Control Rule System Flow 169

CHAPTER 16 **Monitoring a Service Graph** 173
Monitoring a Service Graph Instance Using the GUI 173
Monitoring Service Graph Faults Using the GUI 174
Resolving Service Graph Faults 174
Monitoring a Virtual Device Using the GUI 179
Monitoring Device Cluster and Service Graph Status Using the NX-OS-Style CLI 180

CHAPTER 17 **Configuring Multi-Tier Application with Service Graph** 183
About Multi-Tier Application with Service Graph 183
Creating a Multi-Tier Application Profile Using the GUI 183

CHAPTER 18 **Configuring Administrator Roles for Managing a Service Configuration** 187
About Privileges 187
Configuring a Role for Device Management 188
Configuring a Role for Service Graph Template Management 188
Configuring a Role for Uploading Device Package 188
Configuring a Role for Exporting Devices 188

CHAPTER 19 **Developing Automation** 189

About the REST APIs 189

Examples of Automating Using the REST APIs 189

CHAPTER 20 **Configuring Cloud Orchestrator Mode** 197

About the Cloud Orchestrator Mode 197

Cloud Orchestrator Mode Schema 197

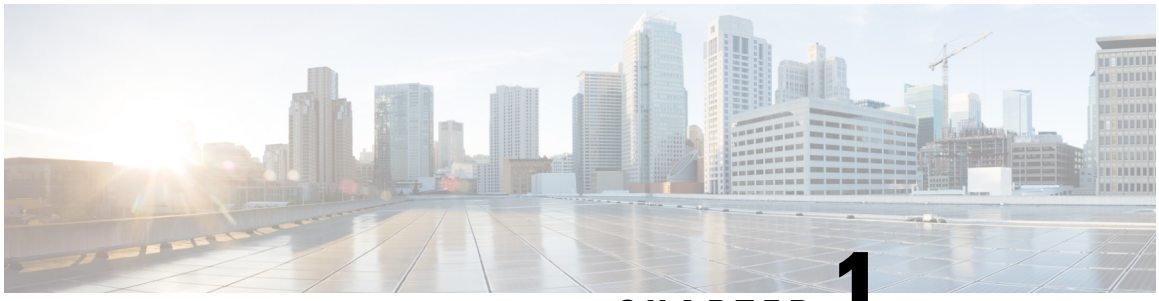
Firewall Schema 197

Load Balancer Schema 201

Configuring the Cloud Orchestrator Mode Using the GUI 204

Configuring a Firewall Using the REST API 205

Configuring a Load Balancer Using the REST API 206



CHAPTER 1

New and Changed Information

• New and Changed Information, on page 1

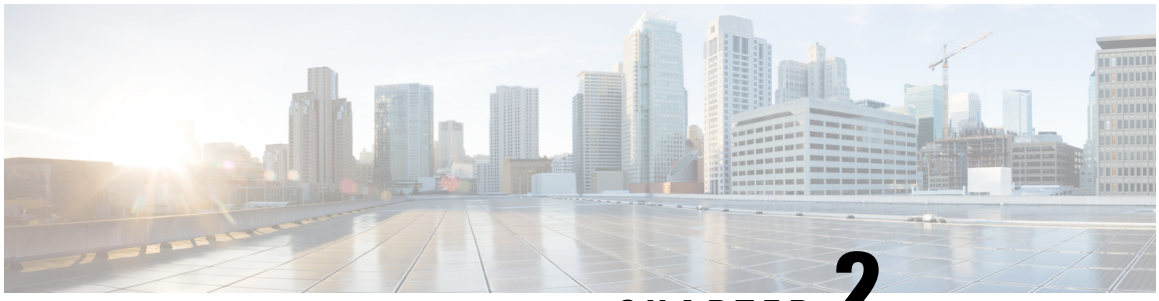
New and Changed Information

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 made to the guide or of the new features up to this release.

Table 1: New Features and Changed Behavior for Cisco APIC Release 3.2(1).

Feature	Description	Where Documented
Multi-Tier Application with Service Graph	Provides a consolidated method of configuring service graph components such as bridge domains, EPGs, VRFs, services, and contracts.	*Configuring a Multi-Tier Application with Service Graph*
Multi-node policy-based redirect (PBR) support	PBR supports up to three nodes in a service chain configured for policy-based redirection.	*Configuring Policy-Based Redirect*
PBR Resilient Hashing	For PBR nodes, enable resilient hashing to avoid rehashing any traffic other than the flows from a disabled or failed node.	*Resilient Hashing*
Service graph support for contracts involving microsegmented EPGs	Layer 4 to Layer 7 service graphs are supported for contracts between microsegmented EPGs and contracts between microsegmented EPGs and regular EPGs.	The chapter "Microsegmentation with Cisco ACI" in the Cisco ACI Virtualization Guide

Feature	Description	Where Documented
Service graph support for Cisco ACI Virtual Edge	Layer 4 to Layer 7 service graphs are supported for Cisco ACI Virtual Edge. Layer 4 to Layer 7 services are supported only for routed mode in the initial release; there is no support for transparent mode.	The chapter "Layer 4 to Layer 7 Services" in the Cisco ACI Virtual Edge Configuration Guide



CHAPTER 2

Overview

- About Deploying Application-Centric Infrastructure Layer 4 to Layer 7 Services, on page 3
- About Layer 4 to Layer 7 Service Devices, on page 4
- About Service Graph Templates, on page 4
- Configuring Layer 4 to Layer 7 Services Using the GUI, on page 5

About Deploying Application-Centric Infrastructure Layer 4 to Layer 7 Services

Traditionally, when you insert services into a network, you must perform a highly manual and complicated VLAN (Layer 2) or virtual routing and forwarding (VRF) instance (Layer 3) stitching between network elements and service appliances. This traditional model requires days or weeks to deploy new services for an application. The services are less flexible, operating errors are more likely, and troubleshooting is more difficult. When an application is retired, removing a service device configuration, such as firewall rules, is difficult. Scale out/scale down of services that is based on the load is also not feasible.

Although VLAN and virtual routing and forwarding (VRF) stitching is supported by traditional service insertion models, the Application Policy Infrastructure Controller (APIC) can automate service insertion while acting as a central point of policy control. The APIC policies manage both the network fabric and services appliances. The APIC can configure the network automatically so that traffic flows through the services. The APIC can also automatically configure the service according to the application's requirements, which allows organizations to automate service insertion and eliminate the challenge of managing the complex techniques of traditional service insertion.

Before you begin, the following APIC objects must be configured:

- The tenant that will provide/consume the Layer 4 to Layer 7 services
- A Layer 3 outside network for the tenant
- At least one bridge domain
- An application profile
- A physical domain or a VMM domain

  For a VMM domain, configure VMM domain credentials and configure a vCenter/vShield controller profile.
- A VLAN pool with an encapsulation block range

- At least one contract
- At least one EPG

You must perform the following tasks to deploy Layer 4 to Layer 7 services:

1. Import a **Device Package** .

   Only the provider administrator can import the device package.
2. Register the device and the logical interfaces.

   This task also registers concrete devices and concrete interfaces, and configures concrete device parameters.
3. Create a **Logical Device**.
4. Configure device parameters.
5. Optional. If you are configuring an ASA Firewall service, enable trunking on the device.
6. Configure a **Device Selection Policy**.
7. Configure a **Service Graph Template**.
   a. Select the default service graph template parameters from an application profile.
   b. Configure additional service graph template parameters, if needed.
8. Attach the service graph template to a contract.
9. Configure additional configuration parameters, if needed.

Note Virtualized appliances can be deployed with VLANs as the transport between VMware ESX servers and leaf nodes, and can be deployed only with VMware ESX as the hypervisor.

About Layer 4 to Layer 7 Service Devices

A Layer 4 to Layer 7 service device is a functional component that is connected to a fabric, such as a firewall, Intrusion-Prevention System (IPS), or load balancer.

About Service Graph Templates

The Cisco Application Centric Infrastructure (ACI) allows you to define a sequence of meta-devices, such a firewall of a certain type followed by a load balancer of a certain make and version. This is called an service graph template, also known as an abstract graph. When a service graph template is referenced by a contract, the service graph template is instantiated by mapping it to concrete devices, such as the firewall and load balancers that are present in the fabric. The mapping happens with the concept of a *context*. The *device context* is the mapping configuration that allows Cisco ACI to identify which firewalls and which load balancers can be mapped to the service graph template. Another key concept is the *logical device*, which represents the cluster of concrete devices. The rendering of the service graph template is based on identifying the suitable logical devices that can be inserted in the path that is defined by a contract.

Cisco ACI treats services as an integral part of an application. Any services that are required are treated as a service graph that is instantiated on the Cisco ACI fabric from the Cisco Application Policy Infrastructure Controller (APIC). Users define the service for the application, while service graph templates identify the set of network or service functions that are needed by the application. Once the graph is configured in the Cisco APIC, the Cisco APIC automatically configures the services according to the service function requirements that are specified in the service graph template. The Cisco APIC also automatically configures the network according to the needs of the service function that is specified in the service graph template, which does not require any change in the service device.

Configuring Layer 4 to Layer 7 Services Using the GUI

The following list provides an overview of how to configure the Layer 4 to Layer 7 services using the GUI:

1. Configure a device.

   See Configuring a Layer 4 to Layer 7 Services Device Using the GUI, on page 13.

   (Optional) Modify a device.

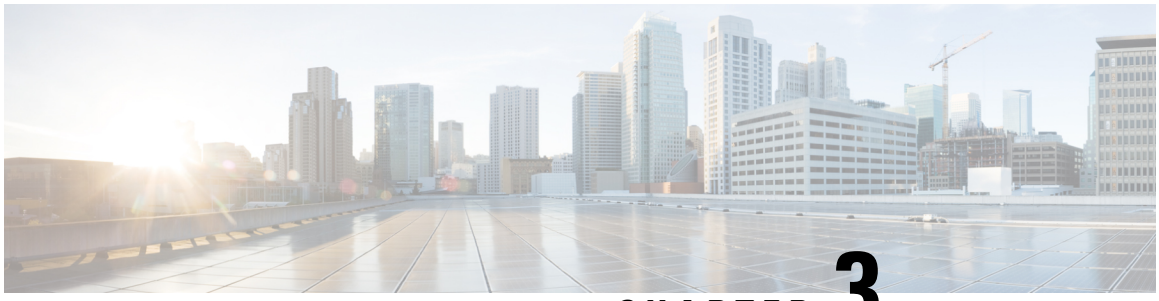
   See Modifying a Device Using the GUI, on page 21.

2. Configure a service graph template.

   See Configuring a Service Graph Template Using the GUI, on page 38.

3. Apply a service graph template to endpoint groups (EPGs).

   See Applying a Service Graph Template to Endpoint Groups Using the GUI, on page 41.



CHAPTER 3

Importing a Device Package

- About Device Packages, on page 7
- Installing a Device Package Using the REST API, on page 9
- Importing a Device Package Using the GUI, on page 10

About Device Packages

The Application Policy Infrastructure Controller (APIC) requires a device package to configure and monitor service devices. You add service functions to the Cisco APIC through the device package. A device package manages a single class of service devices and provides the Cisco APIC with information about the device and its capabilities. A device package is a `zip` file that contains the following parts:

Device specification	An XML file that defines the following:
	• Device properties:
	  • **Model**: Model of the device.
	  • **Vendor**: Vendor of the device.
	  • **Version**: Software version of the device.
	• Functions provided by a device, such as load balancing, content switching, and SSL termination.
	• Interfaces and network connectivity information for each function.
	• Device configuration parameters.
	• Configuration parameters for each function.

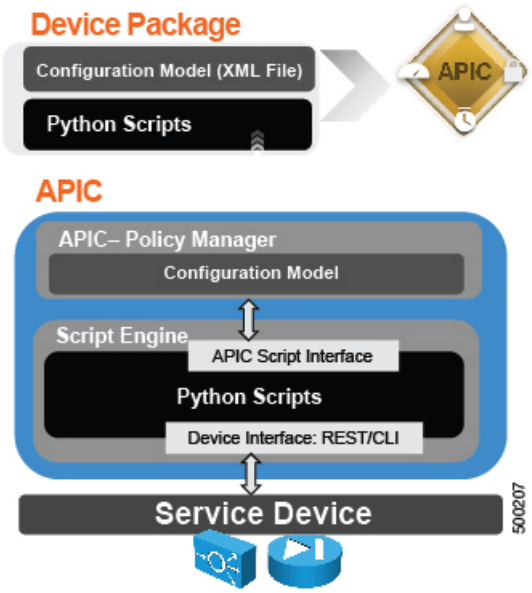
Device script	A Python script that interacts with the device from the Cisco APIC. Cisco APIC events are mapped to function calls that are defined in the device script. A device package can contain multiple device scripts. A device script can interface with the device by using REST, SSH, or any similar mechanism.
Function profile	Function parameters with default values that are specified by the vendor. You can configure a function to use these default values.

Device-level configuration parameters	A configuration file that specifies parameters that are required by a device. This configuration can be shared by one or more graphs using a device.

You can create a device package or it can be provided by a device vendor or Cisco.

The following figure illustrates the interaction of a device package and the Cisco APIC:

Figure 1: Device Package and the Cisco APIC

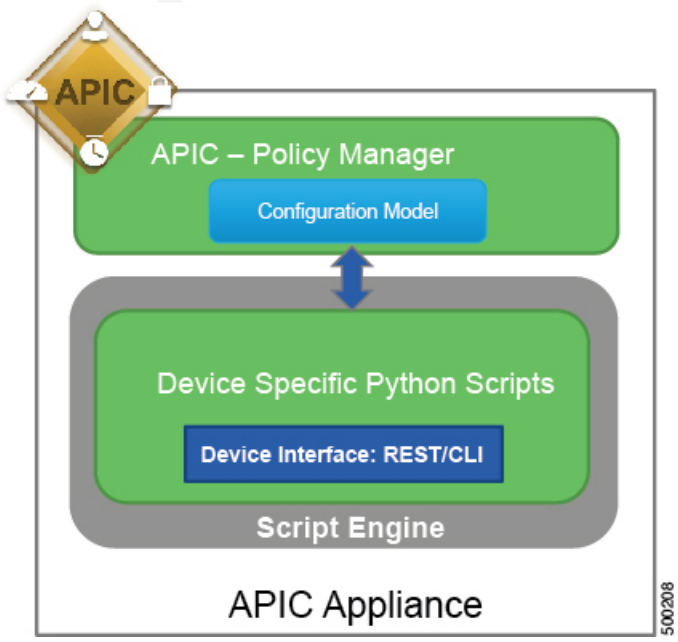


The functions in a device script are classified into the following categories:

- Device/Infrastructure: For device level configuration and monitoring
- Service Events: For configuring functions, such as a server load balancer or Secure Sockets Layer, on the device
- Endpoint/Network Events: For handling endpoint and network attach/detach events

The Cisco APIC uses the device configuration model that is provided in the device package to pass the appropriate configuration to the device scripts. The device script handlers interface with the device using its REST or CLI interface.

Figure 2: How the Device Scripts Interface with a Service Device



The device package enables an administrator to automate the management of the following services:

- Device attachment and detachment
- Endpoint attachment and detachment
- Service graph rendering
- Health monitoring
- Alarms, notifications, and logging
- Counters

For more information about device packages and how to develop a device package, see the *Cisco APIC Layer 4 to Layer 7 Device Package Development Guide*.

Installing a Device Package Using the REST API

You can install a device package using an HTTP or HTTPS POST.

Install the device package.

- If HTTP is enabled on the Application Policy Infrastructure Controller (APIC) , the URL for the POST is as follows:

  ```
  http://10.10.10.10/ppi/node/mo/.xml
  ```

- If HTTPS is enabled on the APIC, the URL for the POST is as follows:

  ```
  https://10.10.10.10/ppi/node/mo/.xml
  ```

The message must have a valid session cookie.

The body of the POST should contain the device package being uploaded. Only one package is allowed in a POST.

Importing a Device Package Using the GUI

Before performing any configuration based on service graphs, you must download and install the appropriate device package in the Application Policy Infrastructure Controller (APIC). A device package specifies to the Cisco APIC what devices you have and what the devices can do.

Note Beginning in the 3.1(1) release, choosing a device package that uses the cloud orchestrator mode provides a simpler interface. The cloud-orchestrator mode device package is automatically created in the Cisco APIC. If the device package is deleted by mistake, it can be uploaded again. See also Configuring Cloud Orchestrator Mode, on page 197.

Step 1 Download an appropriate device package. You can find the list of partners at the following URL:

http://www.cisco.com/c/en/us/solutions/data-center-virtualization/ecosystem.html

This URL is the Partner Ecosystem page, where you can download the appropriate device package.

Step 2 Log in to the Cisco APIC as the provider administrator.

Step 3 On the menu bar, choose **L4-L7 Services** > **Packages**.

Step 4 In the **Navigation** pane, choose **L4-L7 Service Device Types**.

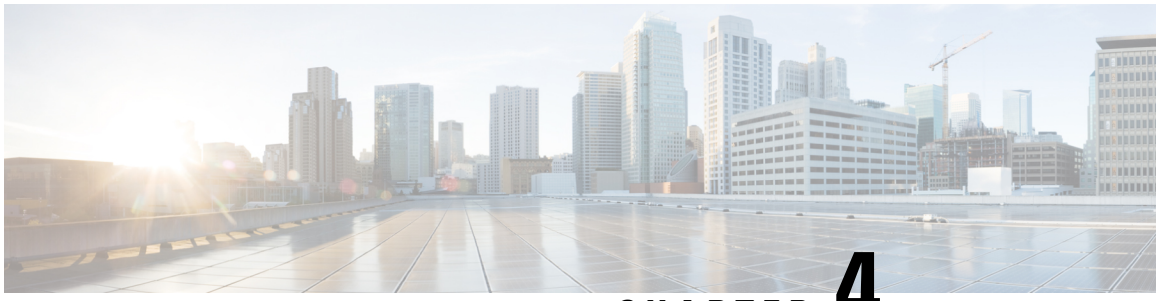
Step 5 In the **Work** pane, choose **Actions** > **Import Device Package**. The **Import Device Package** dialog box appears.

Step 6 Click **Browse...** and browse to the device package that you want to use.

For information about creating device packages, see the *Cisco APIC Layer 4 to Layer 7 Device Package Development Guide*.

Step 7 Click **Open**.

Step 8 Click **Submit**.



CHAPTER 4

Defining a Logical Device

- About Device Clusters, on page 11
- About Concrete Devices, on page 12
- About Trunking, on page 13
- Configuring a Layer 4 to Layer 7 Services Device Using the GUI, on page 13
- Creating a Layer 4 to Layer 7 Device Using the NX-OS-Style CLI, on page 16
- Enabling Trunking on a Layer 4 to Layer 7 Virtual ASA device Using the GUI, on page 20
- Enabling Trunking on a Layer 4 to Layer 7 Virtual ASA device Using the REST APIs, on page 20
- Modifying a Device Using the GUI, on page 21
- Using an Imported Device with the REST APIs, on page 21
- Importing a Device From Another Tenant Using the NX-OS-Style CLI, on page 22
- Verifying the Import of a Device Using the GUI, on page 22

About Device Clusters

A device cluster (also known as a logical device) is one or more concrete devices that act as a single device. A device cluster has cluster (logical) interfaces, which describe the interface information for the device cluster. During service graph template rendering, function node connectors are associated with cluster (logical) interfaces. The Application Policy Infrastructure Controller (APIC) allocates the network resources (VLAN or Virtual Extensible Local Area Network [VXLAN]) for a function node connector during service graph template instantiation and rendering and programs the network resources onto the cluster (logical) interfaces.

The service graph template uses a specific device that is based on a device selection policy (called a *logical device context*) that an administrator defines.

An administrator can set up a maximum of two concrete devices in active-standby mode.

To set up a device cluster, you must perform the following tasks:

1. Connect the concrete devices to the fabric.
2. Assign the management IP address to the device cluster.
3. Register the device cluster with the APIC. The APIC validates the device using the device specifications from the device package.

Note The APIC does not validate a duplicate IP address that is assigned to two device clusters. The APIC can provision the wrong device cluster when two device clusters have the same management IP address. If you have duplicate IP addresses for your device clusters, delete the IP address configuration on one of the devices and ensure there are no duplicate IP addresses that are provisioned for the management IP address configuration.

About Managed Device Clusters

A device cluster can be configured as a managed device cluster. In managed mode, the Application Policy Infrastructure Controller (APIC) programs the devices during graph instantiation using the configuration provided to the APIC by an APIC administrator. For a managed device cluster, the APIC requires the device package for managing the devices in the device cluster.

By default, a device cluster is configured as a managed device cluster.

The following settings are needed when a device cluster is configured as managed:

- Device package
- Connectivity information for the logical device (vnsLDevViP) and devices (CDev)-management IP address, credentials, and in-band connectivity information
- Information about supported function types (go-through, go-to)
- Information about context awareness (single context or multi-context)

The APIC needs to know the topology information (logical interface and concrete interface) for the device cluster and devices. This information is needed so that the APIC can program the appropriate ports on the leaf, and the APIC can also use this information for troubleshooting wizard purposes. The APIC also needs to know the relation to `DomP`, which is used for allocating the encapsulation.

About Unmanaged Device Clusters

A device cluster can be configured as an unmanaged device cluster. For an unmanaged device cluster, the Application Policy Infrastructure Controller (APIC) allocates only the network resources for the service graph and program on only the fabric side during graph instantiation. This might be useful if your environment already has an existing orchestrator or a dev-op tool that programs the devices in a device cluster. In some other cases, the device package for the service appliance is not available. Unmanaged mode enables the APIC to work with service devices without needing to have a device package.

The APIC needs to know the topology information (logical interface and concrete interface) for the device cluster and devices. This information is needed so that the APIC can program the appropriate ports on the leaf, and the APIC can also use this information for troubleshooting wizard purposes. The APIC also needs to know the relation to `DomP`, which is used for allocating the encapsulation.

About Concrete Devices

A concrete device can be either a physical device or a virtual device. A concrete device has concrete interfaces. When a concrete device is added to a logical device, the concrete interfaces are mapped to the logical interfaces.

During service graph template instantiation, VLANs and VXLANs are programmed on concrete interfaces that are based on their association with logical interfaces.

About Trunking

You can enable trunking for a Layer 4 to Layer 7 virtual ASA device, which uses trunk port groups to aggregate the traffic of endpoint groups. Without trunking, a virtual service device can have only 1 VLAN per interface and up to 10 service graphs. With trunking enabled, the virtual service device can have an unlimited number of service graphs.

For more information about trunk port groups, see the *Cisco ACI Virtualization Guide*.

Trunking is supported only on a virtual ASA device. The ASA device package must be version 1.2.7.8 or later.

Configuring a Layer 4 to Layer 7 Services Device Using the GUI

When you create a Layer 4 to Layer 7 services device, you can connect to either a physical device or a virtual machine. The fields are slightly different depending on the type to which you are connecting. When you connect to a physical device, you specify the physical interface. When you connect to a virtual machine, you specify the VMM domain, the virtual machine, and the virtual interfaces. Additionally, you can select an unknown model, which allows you to configure the connections manually.

Note When you configure a Layer 4 to Layer 7 services device that is a load balancer, the context aware parameter is not used. The context aware parameter has a default value of `single context`, which can be ignored.

Before you begin

- You must have configured a tenant.

Step 1 On the menu bar, choose **Tenants** > **All Tenants**.

Step 2 In the Work pane, double click the tenant's name.

Step 3 In the Navigation pane, choose **Tenant** ***tenant_name*** > **Services** > **L4-L7** > **Devices**.

Step 4 In the Work pane, choose **Actions** > **Create L4-L7 Devices**.

Step 5 In the **Create L4-L7 Devices** dialog box, in the **General** section, complete the following fields:

Name	Description
Managed check box	Put a check in the box to create a managed device, or remove the check from the box to create an unmanaged device.
Name field	Enter a name for the device.

Name	Description
Service Type drop-down list	Choose the service type. The types are: • ADC • Firewall • Other **Note** For L1/L2 Firewall configuration use **Other**.
Device Type buttons	Choose the device type.
Physical Domain or **VMM Domain** drop-down list	Choose the physical domain or VMM domain.
Switching Mode (Cisco ACI Virtual Edge only)	For a Cisco ACI Virtual Edge virtual domain, choose one of the following modes: • **AVE**—Traffic is switched through the Cisco ACI Virtual Edge. • **native**—Traffic is switched through the VMware DVS.
View radio buttons	Choose the view for the device. The view can be: • **Single Node**—Only one node • **HA Node**—High availability nodes (two nodes) • **Cluster**—3 or more nodes
Device Package drop-down list	(Only for managed devices) Choose the vendor-provided device package that you will use.
Model drop-down list	(Only for managed devices) Choose the model of the device.
Context Aware	The context awareness of the device. The awareness can be: • **Single**: The device cluster cannot be shared across multiple tenants of a given type that are hosted on the provider network. You must give the device cluster to a specific tenant for a given user. • **Multiple**: The device cluster can be shared across multiple tenants of a given type that you are hosting on the provider network. For example, there could be two hosting companies that share the same device. The default is **Single**. **Note** When you create a Layer 4 to Layer 7 services device that is a load balancer, the **Context Aware** parameter is not used and can be ignored.

Name	Description
Function Type	Function types are: • **GoThrough**: Transparent mode • **GoTo**: Routed mode • **L1**: L1 Firewall mode • **L2**: L2 Firewall mode The default is **GoTo**.

Step 6 (Only for managed devices) In the **Connectivity** section, complete the following fields:

Name	Description
APIC to Device Management Connectivity radio buttons	Choose the type of connectivity. Choose **Out-of-Band** when you are connecting to a device that is outside of the fabric or **In-Band** when you are connecting to a device through the fabric.

Step 7 (Only for managed devices) In the **Credentials** section, complete the following fields:

Name	Description
User Name field	Enter your user name.
Password field	Enter your password.
Confirm Password field	Enter your password again.

Step 8 In the **Device 1** section, complete the following fields:

Name	Description
Management IP Address field	(Only for managed devices) Enter the management IP address of the device to which you are connecting.
Management Port field and drop-down list	(Only for managed devices) Enter the management port or choose a value from the drop-down list.
VM drop-down list	(Only for the virtual device type) Choose a virtual machine.
Chassis drop-down list	(Only for managed devices) Choose a chassis.

Step 9 In the **Device Interfaces** table, click the **+** button to add an interface and complete the following fields:

Name	Description
Name drop-down list	Choose the interface name.
VNIC drop-down list	(Only for the virtual device type) Choose a vNIC.

Name	Description
Path drop-down list	Choose a port, port channel, or virtual port channel to which the interface will connect.

Step 10 Click **Update**.

Step 11 (Only for an HA cluster) Complete the fields for each device.

Step 12 Complete the fields for the **Cluster** section.

For an HA cluster, make sure that the cluster interfaces are mapped to the corresponding interfaces on both concrete devices in the cluster.

Step 13 Click **Next**.

The **Device Configuration** page displays a list of possible features and parameters for the package you are using. You see a tab with the **Basic** parameters displayed and another tab **All Parameters** that displays all the available parameters with your device package. The basic parameters are included under **All Parameters**.

Step 14 In the **Features** section, choose the set of features that you want to use.
The set of parameters changes depending on the specific package you are using and the specific feature you select.

Step 15 For the parameters of the chosen features, supply the values as follows:

a) Double-click in the field you want to modify.
b) Enter the required information in the fields that appear.
c) Click **Update**.

Step 16 Click **Finish**.

Creating a Layer 4 to Layer 7 Device Using the NX-OS-Style CLI

When you create a Layer 4 to Layer 7 device, you can connect to either a physical device or a virtual machine. When you connecting to a physical device, you specify the physical interface. When you connect to a virtual machine, you specify the VMM domain, the virtual machine, and the virtual interfaces.

Note When you configure a Layer 4 to Layer 7 device that is a load balancer, the context aware parameter is not used. The context aware parameter has a default value of `single context`, which can be ignored.

Before you begin

- You must have configured a tenant.

Step 1 Enter the configure mode.

Example:

```
apic1# configure
```

Step 2 Enter the configure mode for a tenant.

```
tenant tenant_name
```

Example:

```
apic1(config)# tenant t1
```

Step 3 Add a Layer 4 to Layer 7 device cluster.

```
l4l7 cluster name cluster_name type cluster_type vlan-domain domain_name
  [function function_type] [service service_type]
```

Parameter	Description
name	The name of the device cluster.
type	The type of the device cluster. Possible values are: • virtual • physical
vlan-domain	The domain to use for allocating the VLANs. The domain must be a VMM domain for virtual device and physical domain for physical device.
switching-mode (Cisco ACI Virtual Edge only)	(Optional) Choose one of the following modes: • **AVE**—Switches traffic through the Cisco ACI Virtual Edge. • **native**—Switches traffic through the VMware DVS. This is the default value.
function	(Optional) The function type. Possible values are: • go-to • go-through • L1 • L2
service	(Optional) The service type. This is used by the GUI to show the ADC- or firewall-specific icons and GUI. Possible values are: • ADC • FW • OTHERS

Example:

For a physical device, enter:

```
apic1(config-tenant)# l4l7 cluster name D1 type physical vlan-domain phys
  function go-through service ADC
```

For a virtual device, enter:

```
apic1(config-tenant)# l4l7 cluster name ADCCluster1 type virtual vlan-domain mininet
```

Step 4 Add one or more cluster devices in the device cluster.

```
cluster-device device_name [vcenter vcenter_name] [vm vm_name]
```

Parameter	Description
vcenter	(Only for a virtual device) The name of VCenter that hosts the virtual machine for the virtual device.
vm	(Only for a virtual device) The name of the virtual machine for the virtual device.

Example:

For a physical device, enter:

```
apic1(config-cluster)# cluster-device C1
apic1(config-cluster)# cluster-device C2
```

For a virtual device, enter:

```
apic1(config-cluster)# cluster-device C1 vcenter vcenter1 vm VM1
apic1(config-cluster)# cluster-device C2 vcenter vcenter1 vm VM2
```

Step 5 Add one or more cluster interfaces in the device cluster.

```
cluster-interface interface_name [vlan static_encap]
```

Parameter	Description
vlan	(Only for a physical device) The static encapsulation for the cluster interface. VLAN value must be between 1 to 4094.

Example:

For a physical device, enter:

```
apic1(config-cluster)# cluster-interface consumer vlan 1001
```

For a virtual device, enter:

```
apic1(config-cluster)# cluster-interface consumer
```

Step 6 Add one or more members in the cluster interface.

```
member device device_name device-interface interface_name
```

Parameter	Description
device	The name of the cluster device that must have been already added to this device cluster using **cluster-device** command.
device-interface	The name of the interface on the cluster device.

Example:

```
apic1(config-cluster-interface)# member device C1 device-interface 1.1
```

Step 7 Add an interface to a member.

```
interface {ethernet ethernet_port | port-channel port_channel_name [fex fex_ID] |
  vpc vpc_name [fex fex_ID]} leaf leaf_ID
```

If you want to add a vNIC instead of an interface, then skip this step.

Parameter	Description
ethernet	(Only for an Ethernet or FEX Ethernet interface) The Ethernet port on the leaf where the cluster device is connected to Cisco Application Centric Infrastructure (ACI) fabric. If you are adding a FEX Ethernet member, specify both the FEX ID and the FEX port in the following format: *FEX_ID/FEX_port* For example: 101/1/23 The FEX ID specifies where the cluster device is connected to Fabric extender.
port-channel	(Only for a port channel or FEX port channel interface) The port channel name where the cluster device is connected to ACI fabric.
vpc	(Only for a virtual port channel or FEX virtual port channel interface) The virtual port channel name where the cluster device is connected to ACI fabric.
fex	(Only for a port channel, FEX port channel, virtual port channel, or FEX virtual port channel interface) The FEX IDs in a space-separated list that are used to form the port channel or virtual port channel.
leaf	The leaf IDs in a space-separated list where the cluster device is connected.

Example:

For an Ethernet interface, enter:

```
apic1(config-member)# interface ethernet 1/23 leaf 101
apic1(config-member)# exit
```

For a FEX Ethernet interface, enter:

```
apic1(config-member)# interface ethernet 101/1/23 leaf 101
apic1(config-member)# exit
```

For a port channel interface, enter:

```
apic1(config-member)# interface port-channel pc1 leaf 101
apic1(config-member)# exit
```

For a FEX port channel interface, enter:

```
apic1(config-member)# interface port-channel pc1 leaf 101 fex 101
apic1(config-member)# exit
```

For a virtual port channel interface, enter:

```
apic1(config-member)# interface vpc vpc1 leaf 101 102
apic1(config-member)# exit
```

For a FEX virtual port channel interface, enter:

```
apic1(config-member)# interface vpc vpc1 leaf 101 102 fex 101 102
apic1(config-member)# exit
```

Step 8 Add a vNIC to a member.

```
vnic "vnic_name"
```

If you want to add an interface instead of a vNIC, then see the previous step.

Parameter	Description
vnic	The name of the vNIC adapter on the virtual machine for the cluster-device. Enclose the name in double quotes.

Example:

```
apic1(config-member)# vnic "Network adapter 2"
apic1(config-member)# exit
```

Step 9 If you are done creating the device, exit the configuration mode.

Example:

```
apic1(config-cluster-interface)# exit
apic1(config-cluster)# exit
apic1(config-tenant)# exit
apic1(config)# exit
```

Enabling Trunking on a Layer 4 to Layer 7 Virtual ASA device Using the GUI

The following procedure enables trunking on a Layer 4 to Layer 7 virtual ASA device using the GUI.

Before you begin

- You must have configured a Layer 4 to Layer 7 virtual ASA device.

Step 1 On the menu bar, choose **Tenants** > **All Tenants**.

Step 2 In the Work pane, double click the tenant's name.

Step 3 In the Navigation pane, choose **Tenant** ***tenant_name*** > **Services** > **L4-L7** > **Devices** > ***device_name*** .

Step 4 In the Work pane, put a check in the **Trunking Port** check box.

Step 5 Click **Submit**.

Enabling Trunking on a Layer 4 to Layer 7 Virtual ASA device Using the REST APIs

The following procedure provides an example of enabling trunking on a Layer 4 to Layer 7 virtual ASA device using the REST APIs.

Before you begin

- You must have configured a Layer 4 to Layer 7 virtual ASA device.

Enable trunking on the Layer 4 to Layer 7 device named `InsiemeCluster`:

```
<polUni>
    <fvTenant name="tenant1">
        <vnsLDevVip name="InsiemeCluster" devtype="VIRTUAL" trunking="yes">
            ...
            ...
        </vnsLDevVip>
    </fvTenant>
</polUni>
```

Modifying a Device Using the GUI

After you create a device, you can modify the device.

Note To create a device or to add a device to an existing cluster, you must use the "Creating a Device" procedure.

Step 1 On the menu bar, choose **Tenants** > **All Tenants**.

Step 2 In the Work pane, double click the tenant's name.

Step 3 In the Navigation pane, choose **Tenant** ***tenant_name*** > **Services** > **L4-L7** > **Devices** > ***device_name***.
The Work pane displays information about the device.

Step 4 You can change some of the parameters in the **General** section.

You can add interfaces or change the path for the existing interfaces in the **Device 1** section. To add an interface, click the **+** button. To change the path, double-click on the path you want to change.

Step 5 After you making any changes to the parameters, click **Submit**.

Using an Imported Device with the REST APIs

The following REST API uses an imported device:

```
<polUni>
  <fvTenant dn="uni/tn-tenant1" name="tenant1">
    <vnsLDevIf ldev="uni/tn-mgmt/lDevVip-ADCCluster1"/>
    <vnsLDevCtx ctrctNameOrLbl="any" graphNameOrLbl="any" nodeNameOrLbl="any">
      <vnsRsLDevCtxToLDev tDn="uni/tn-tenant1/lDevIf-[uni/tn-mgmt/lDevVip-ADCCluster1]"/>
      <vnsLIfCtx connNameOrLbl="inside">
        <vnsRsLIfCtxToLIf
tDn="uni/tn-tenant1/lDevIf-[uni/tn-mgmt/lDevVip-ADCCluster1]/lDevIfLIf-inside"/>
        <fvSubnet ip="10.10.10.10/24"/>
```

```
        <vnsRsLIfCtxToBD tDn="uni/tn-tenant1/BD-tenant1BD1"/>
      </vnsLIfCtx>
      <vnsLIfCtx connNameOrLbl="outside">
        <vnsRsLIfCtxToLIf
tDn="uni/tn-tenant1/lDevIf-[uni/tn-mgmt/lDevVip-ADCCluster1]/lDevIfLIf-outside"/>
        <fvSubnet ip="70.70.70.70/24"/>
        <vnsRsLIfCtxToBD tDn="uni/tn-tenant1/BD-tenant1BD4"/>
      </vnsLIfCtx>
    </vnsLDevCtx>
  </fvTenant>
</polUni>
```

Importing a Device From Another Tenant Using the NX-OS-Style CLI

You can import a device from another tenant for a shared services scenario.

Step 1 Enter the configure mode.

Example:

```
apic1# configure
```

Step 2 Enter the configure mode for a tenant.

```
tenant tenant_name
```

Example:

```
apic1(config)# tenant t1
```

Step 3 Import the device.

```
l4l7 cluster import-from tenant_name device-cluster device_name
```

Parameter	Description
import-from	Name of the tenant from where to import the device.
device-cluster	Name of the device cluster to import from the specified tenant.

Example:

```
apic1(config-tenant)# l4l7 cluster import-from common device-cluster d1
apic1(config-import-from)# end
```

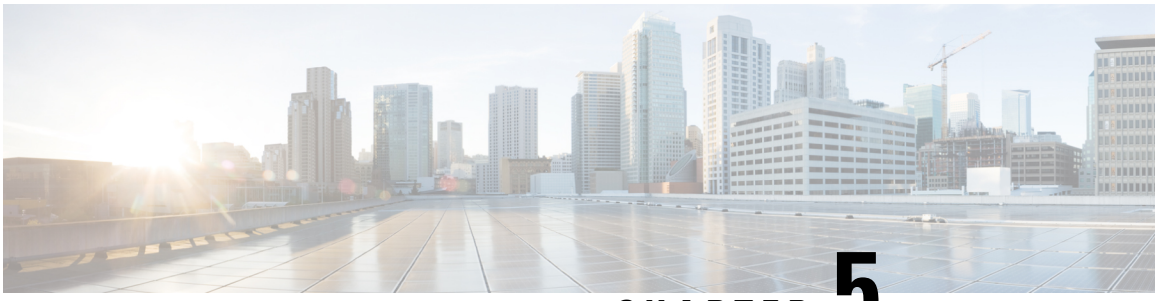
Verifying the Import of a Device Using the GUI

You can use the GUI to verify that a device was imported successfully.

Step 1 On the menu bar, choose **Tenants** > **All Tenants**.

Step 2 In the Work pane, double click the tenant's name.

Step 3 In the Navigation pane, choose **Tenant** ***tenant_name*** > **Services** > **L4-L7** > **Imported Devices** > ***device_name*** .
The device information appears in the **Work** pane.



CHAPTER 5

Configuring Connectivity to Devices

- About In-Band Management for Devices, on page 25
- Configuring In-Band Management for Devices Using the GUI, on page 26
- Troubleshooting In-Band Management for Devices Using the GUI, on page 27

About In-Band Management for Devices

The Cisco Application Policy Infrastructure Controller (Cisco APIC) provides a mechanism for managing devices within each tenant in-band through the Cisco Application Centric Infrastructure (ACI) fabric. This configuration option provides device management connectivity without requiring the management IP addresses used on devices to be routable within the infra tenant and mgmt tenant.

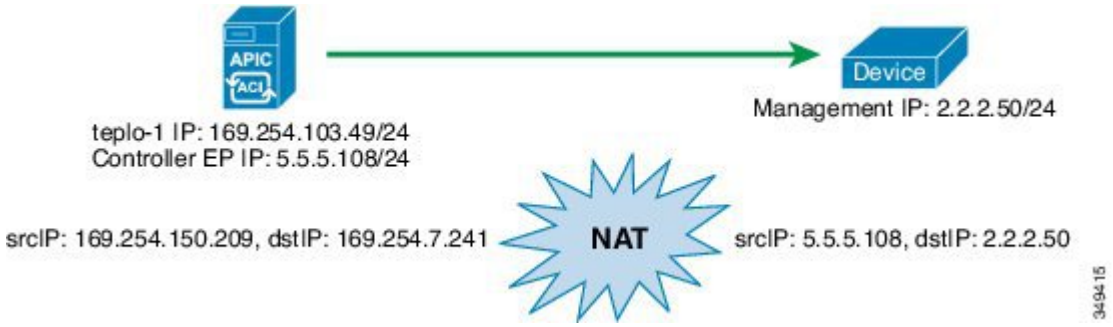
Note This feature is separate from in-band management for the Cisco APICs and fabric nodes. In-band management for the fabric is not required for you to manage devices in-band.

In-band management communication between the Cisco APICs and devices is enabled by configuring unique IP addresses on the Cisco APICs. The IP addresses are known as controller endpoints. These IP addresses are not actually configured on the Cisco APIC interfaces, but instead are used in conjunction with Network Address Translation (NAT) to establish management communication with the devices. The NAT addresses that are used by the Cisco APICs are automatically selected by the Cisco APIC and fall within the 169.254.0.0/16 address range.

In addition, each device management IP address is presented to the Cisco APICs as a translated IP address. This translated address is referred to as the mapped host address.

The following figure depicts the address translation between the Cisco APIC and the devices:

Figure 3: Network Address Translation Between the Cisco APIC and the Devices



Configuring In-Band Management for Devices Using the GUI

You can configure in-band management for devices using the GUI.

Step 1 On the menu bar, choose **Tenants** > **All Tenants**.

Step 2 In the Work pane, double click the tenant's name.

Step 3 In the Navigation pane, choose **Tenant** *tenant_name* > **Services** > **L4-L7** > **Devices**.

Step 4 In the Work pane, choose **Actions** > **Create L4-L7 Devices**

Step 5 In the **Create L4-L7 Devices** dialog box, fill in the fields as required, except as specified below:

a) For the **APIC to Device Management Connectivity** radio buttons, choose **In-Band**.

b) In the **EPG** drop-down list, choose **Create Management EPG**.

Step 6 In the **Create Management EPG** dialog box, fill in the fields as required, except as specified below:

a) In the **Application Profile** drop-down list, choose an existing application profile where the EPG will reside. Optionally, create a new application profile by choosing **Create Application Profile**.

If you create a new application profile, leave the **EPG** section and **Contracts** section blank.

b) In the **Name** field, enter a name for the management EPG.

c) In the **Bridge Domain** drop-down list, choose a domain.

d) In the **Domains** section, add a domain profile.

e) In the **Reserved IP addresses for APICs** section, click + to create a new IP address pool.

Step 7 In the **Create IP Address Pool** dialog box, fill in all of the fields and click **OK**.

The IP address pool defines the controller endpoint addresses. The IP addresses in the pool are the IP addresses that the devices will see as the Application Policy Infrastructure Controller (APIC) IP addresses.

If the address range that you defined for the controller endpoints is not part of the same subnet as the management IP addresses that you defined for the devices, you must define a subnet under the management EPG bridge domain that provides a next-hop gateway for the devices to reach the controller endpoints.

Step 8 In the **Create Management EPG** dialog box, click **Submit**.
The domain name for the management EPG should now be populated.

Step 9 In the **Create L4-L7 Devices** dialog box, complete the device setup. Be sure to include the management interface in the configuration of the interfaces.

Troubleshooting In-Band Management for Devices Using the GUI

If you chose an existing endpoint group (EPG) as the management EPG for the devices, you must manually add the management IP address pools and controller management policies. You can add these using the GUI.

Step 1 On the menu bar, choose **Tenants** > **All Tenants**.

Step 2 In the Work pane, double click the tenant's name.

Step 3 In the Navigation pane, choose ***tenant_name*** > **Application Profiles** > ***application_profile_name*** > **Application EPGs** > ***EPG_name*** > **L4/L7 IP Address Pool**.

Step 4 In the Work pane, choose **Actions** > **Create Address Pool**.

Step 5 In the **Create IP Address Pool** dialog box, fill in the fields as required.

This adds the management IP address pool.

Step 6 In the Navigation pane, choose **Tenant** ***tenant_name*** > **Services** > **L4-L7** > **Inband Management Configuration for L4-L7 devices**.

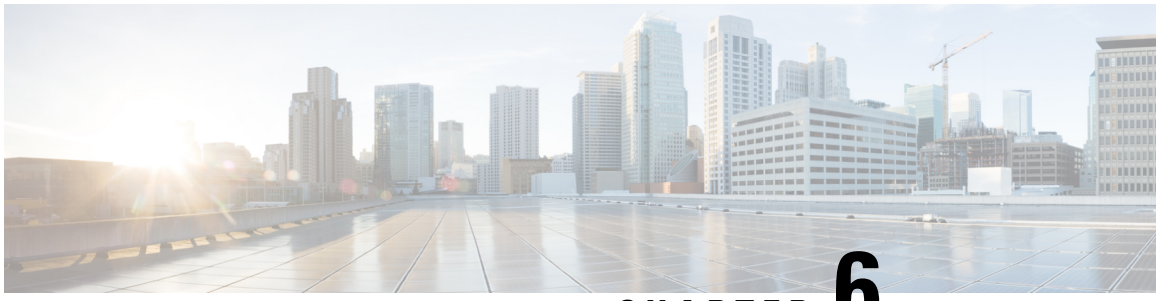
Step 7 In the Work pane, in the **Controller Management Policies** section, click **+** and fill in the fields as follows:

a) In the **Private Networks** drop-down list, choose a private network.

b) In the **Address Pool** drop-down list, choose the pool that you just created.

Step 8 Click **Update**.

This adds the controller management policy.



CHAPTER 6

Selecting a Layer 4 to Layer 7 Device to Render a Graph

- About Device Selection Policies, on page 29
- Creating a Device Selection Policy Using the GUI, on page 29
- Configuring a Device Selection Policy Using REST APIs, on page 32

About Device Selection Policies

A device can be selected based on a contract name, a graph name, or the function node name inside the graph. After you create a device, you can create a device context, which provides a selection criteria policy for a device.

A device selection policy (also known as a device context) specifies the policy for selecting a device for a service graph template. This allows an administrator to have multiple device and then be able to use them for different service graph templates. For example, an administrator can have a device that has high-performance ADC appliances and another device that has lower-performance ADC appliances. Using two different device selection policies, one for the high-performance ADC device and the other for the low-performance ADC device, the administrator can select the high-performance ADC device for the applications that require higher performance and select the low-performance ADC devices for the applications that require lower performance.

Creating a Device Selection Policy Using the GUI

If you did not use the **Apply L4-L7 Service Graph Template To EPGs** wizard to apply the service graph template, you might need to configure a device selection policy (also known as a logical device context). The device selection policy instructs Cisco Application Centric Infrastructure (ACI) about which firewall or load balancer device to use to render a graph.

If you used the **Apply L4-L7 Service Graph Template To EPGs** wizard to apply the service graph template, then a device selection policy was configured automatically and you do not need to configure one manually.

The context name in device selection policy needs to be configured if the device cluster interface is used for intra-vrf and inter-vrf contract. The context name shall be identical for the same device shared by different deployed graph instances.

For example, when you have contract1 that is for intra-vrf and contract2 that is for inter-vrf traffic, if both the contracts have service graph, and you use same device cluster interface, you should configure same context name in device selection policy.

Note When using the NX-OS-style CLI, the device selection policy is configured automatically; there are no equivalent NX-OS-style CLI commands.

If you add copy devices to a service graph template that is already deployed, you must create a device selection policy to use for copy services.

Step 1 On the menu bar, choose **Tenants** > **All Tenants**.

Step 2 In the Work pane, double click the tenant's name.

Step 3 In the Navigation pane, choose **Tenant** *tenant_name* > **Services** > **L4-L7** > **Devices Selection Policies**.

Step 4 In the Work pane, choose **Actions** > **Create Logical Device Context**.

Step 5 In the **Create Logical Device Context** dialog box, fill in the fields as required, except as specified below:

a) In the **Contract Name** drop-down list, choose the contract for the device selection policy. If you do not want to use the contract name as part of the criteria for using a device, choose **any**.

b) In the **Graph Name** drop-down list, choose the graph for the device selection policy. If you do not want to use the graph name as part of the criteria for using a device, choose **any**.

c) In the **Node Name** drop-down list, choose the node for the device selection policy. If you do not want to use the node name as part of the criteria for using a device, choose **any**.

Step 6 In the **Cluster Interface Contexts** section, click **+** to add a cluster interface context.

Property	Description
Connector Name	The connector name or label for the logical interface context. The default is **Any**.
Logical Interface	The logical interface identifier.
Bridge Domain	The private Layer 2 bridge domain consisting of a set of physical or virtual ports. For a copy device, do not choose a bridge domain; the bridge domain is created internally.
L3 Network	The Layer 3 context name. For a copy device, do not choose a Layer 3 network.
L4-L7 Policy based Routing	The policy-based redirect policy to use with logical device context. For a copy device, do not choose a policy-based redirect policy.
Permit Logging	The permit logging status for the interface context. The default is **false**.

Step 7 In the **Create A Cluster Interface Context** dialog, configure the following properties:

Property	Description
Connector Name	The connector name or label for the logical interface context. The default is **Any**.
Cluster Interface	The unique name of the target interface. **Note** This field is required.

Property	Description
Bridge Domain	Enter the associated network of the target. For Anycast, the bridge domain should be the same as that used for the node. **Note** The associated network of the target should be *either* a bridge domain *or* an L3 network.
L3 Network	Enter the associated network of the target. **Note** The associated network of the target should be *either* a bridge domain *or* an L3 network.
L3 Destination (VIP)	Indicates whether this logical interface is terminating the L3 traffic in the service chain. The default for this parameter is enabled (checked). However, this setting is not considered if a policy-based redirect policy is configured on logical interface context. **Note** For Multi-Node PBR, if this logical interface is a consumer construct on a load balancer terminated on a virtual IP external network, put a check in this box and remove any association to a redirect policy in the next field (**L4-L7 Policy Based Redirect**). If this logical interface is a provider construct on a load balancer and it is performing SNAT, then put a check in this box and remove any association to a redirect policy in the next field (**L4-L7 Policy Based Redirect**).
L4-L7 Policy Based Redirect	Optional. Choose the Policy Based Redirect policy or **Create L4-L7 Policy Based Redirect**. **Note** For Multi-Node PBR, if this logical interface is a consumer construct on a load balancer terminated on a virtual IP external network, remove this association to a redirect policy (if entered) and put a check in the **L3 Destination (VIP)** box.
Custom QoS Policy	Optional. Choose a Custom QoS Policy, the default policy, or **Create Custom QoS Policy**.

Property	Description
Preferred Contract Group	The preferred group policy enforcement type. Valid types: • **Include**: EPGs or interfaces configured with this policy option are included in the subgroup and can communicate with others in the subgroup without a contract. • **Exclude**: EPGs or interfaces configured with this policy option are not included in the subgroup and cannot communicate with others in the subgroup without a contract.
Permit Logging	Enable permit logging for the interface context. Default is disabled (false).
Subnets	Click + to add a subnet. Configure the gateway address, the network visibility of the subnet (scope), primary IP address (preferred subnet), and the subnet control state.
Virtual IP Addresses	Click + to add a Virtual IP Address (VIP) if this subnet is used for an L3 virtual destination (**L3 Destination (VIP)** is checked).

Step 8 Click **OK**.

Step 9 Click **Submit**.

Configuring a Device Selection Policy Using REST APIs

You can use the REST APIs to configure a device selection policy.

Creating a Device Selection Policy Using the REST API

The following REST API creates a device selection policy:

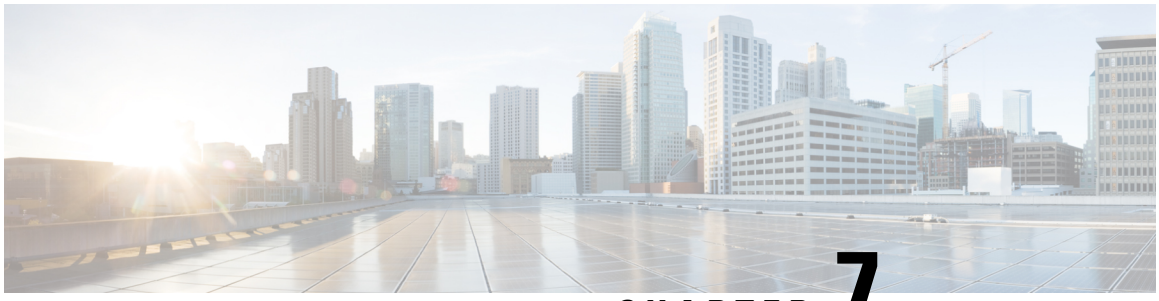
```
<polUni>
    <fvTenant dn="uni/tn-acme" name="acme">
        <vnsLDevCtx ctrctNameOrLbl="webCtrct" graphNameOrLbl="G1" nodeNameOrLbl="Node1">
            <vnsRsLDevCtxToLDev tDn="uni/tn-acme/lDevVip-ADCCluster1"/>

             <!-- The connector name C4, C5, etc.. should match the
                  Function connector name used in the service graph template -->

             <vnsLIfCtx connNameOrLbl="C4">
                <vnsRsLIfCtxToLIf tDn="uni/tn-acme/lDevVip-ADCCluster1/LIf-ext"/>
            </vnsLIfCtx>
            <vnsLIfCtx connNameOrLbl="C5">
                <vnsRsLIfCtxToLIf tDn="uni/tn-acme/lDevVip-ADCCluster1/LIf-int"/>
            </vnsLIfCtx>
```

```
        </vnsLDevCtx>
    </fvTenant>
</polUni>
```

Adding a Logical Interface in a Device Using the REST APIs

The following REST API adds a logical interface in a device:

```
<polUni>
    <fvTenant dn="uni/tn-acme" name="acme">
        <vnsLDevVip name="ADCCluster1">

            <!-- The LIF name defined here (such as e.g., ext, or int) should match the
                 vnsRsLIfCtxToLIf 'tDn' defined in LifCtx  -->

            <vnsLIf name="ext">

                <vnsRsMetaIf tDn="uni/infra/mDev-Acme-ADC-1.0/mIfLbl-outside"/>
                <vnsRsCIfAtt tDn="uni/tn-acme/lDevVip-ADCCluster1/cDev-ADC1/cIf-ext"/>
            </vnsLIf>
            <vnsLIf name="int">
                <vnsRsMetaIf tDn="uni/infra/mDev-Acme-ADC-1.0/mIfLbl-inside"/>
                <vnsRsCIfAtt tDn="uni/tn-acme/lDevVip-ADCCluster1/cDev-ADC1/cIf-int"/>
            </vnsLIf>
        </vnsLDevVip>
    </fvTenant>
</polUni>
```



CHAPTER 7

Configuring a Service Graph

- About Service Graphs, on page 35
- About Function Nodes, on page 37
- About Function Node Connectors, on page 37
- About Service Graph Connections, on page 38
- About Terminal Nodes, on page 38
- Guidelines and Limitations for Configuring Service Graph, on page 38
- About Service Graph Template Configuration Parameters, on page 38
- Configuring a Service Graph Template Using the GUI, on page 38
- Configuring a Service Graph Template Using the REST APIs, on page 40
- Applying a Service Graph Template to Endpoint Groups Using the GUI, on page 41
- Associating a Service Graph Template with a Contract Using the NX-OS-Style CLI, on page 42
- Creating a Function Profile Using the GUI, on page 46
- Using an Existing Function Profile to Create a New Function Profile Using the GUI, on page 47

About Service Graphs

The Cisco Application Centric Infrastructure (ACI) treats services as an integral part of an application. Any services that are required are treated as a service graph that is instantiated on the ACI fabric from the Cisco Application Policy Infrastructure Controller (APIC). Users define the service for the application, while service graphs identify the set of network or service functions that are needed by the application.

A service graph represents the network using the following elements:

- Function node—A function node represents a function that is applied to the traffic, such as a transform (SSL termination, VPN gateway), filter (firewalls), or terminal (intrusion detection systems). A function within the service graph might require one or more parameters and have one or more connectors.
- Terminal node—A terminal node enables input and output from the service graph.
- Connector—A connector enables input and output from a node.
- Connection—A connection determines how traffic is forwarded through the network.

After the graph is configured in the APIC, the APIC automatically configures the services according to the service function requirements that are specified in the service graph. The APIC also automatically configures the network according to the needs of the service function that is specified in the service graph, which does not require any change in the service device.

A service graph is represented as two or more tiers of an application with the appropriate service function inserted between.

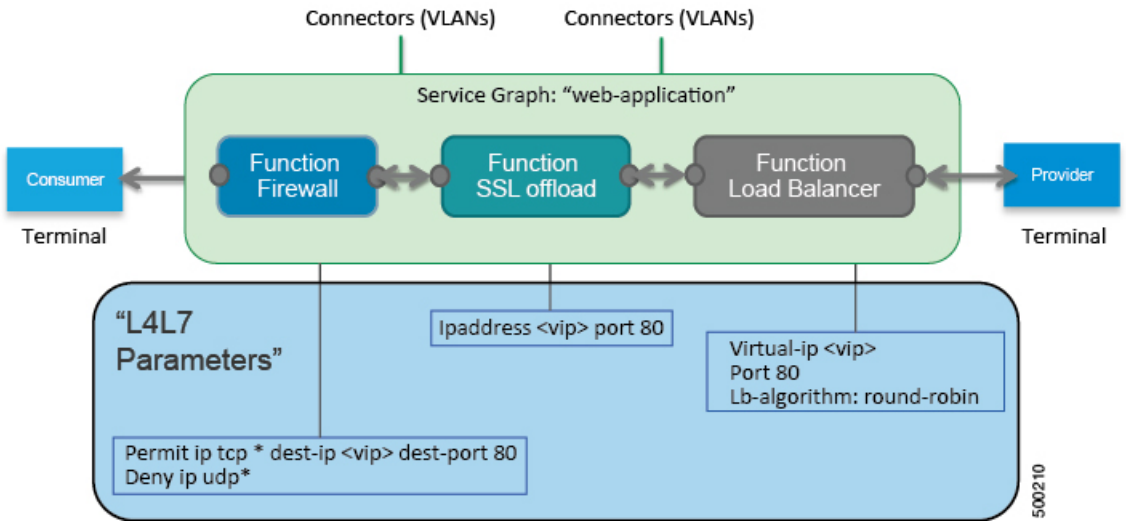
A service appliance (device) performs a service function within the graph. One or more service appliances might be required to render the services required by a graph. One or more service functions can be performed by a single-service device.

Service graphs and service functions have the following characteristics:

- Traffic sent or received by an endpoint group can be filtered based on a policy, and a subset of the traffic can be redirected to different edges in the graph.
- Service graph edges are directional.
- Taps (hardware-based packet copy service) can be attached to different points in the service graph.
- Logical functions can be rendered on the appropriate (physical or virtual) device, based on the policy.
- The service graph supports splits and joins of edges, and it does not restrict the administrator to linear service chains.
- Traffic can be reclassified again in the network after a service appliance emits it.
- Logical service functions can be scaled up or down or can be deployed in a cluster mode or 1:1 active-standby high-availability mode, depending on the requirements.

The following figure provides an example of a service graph deployment:

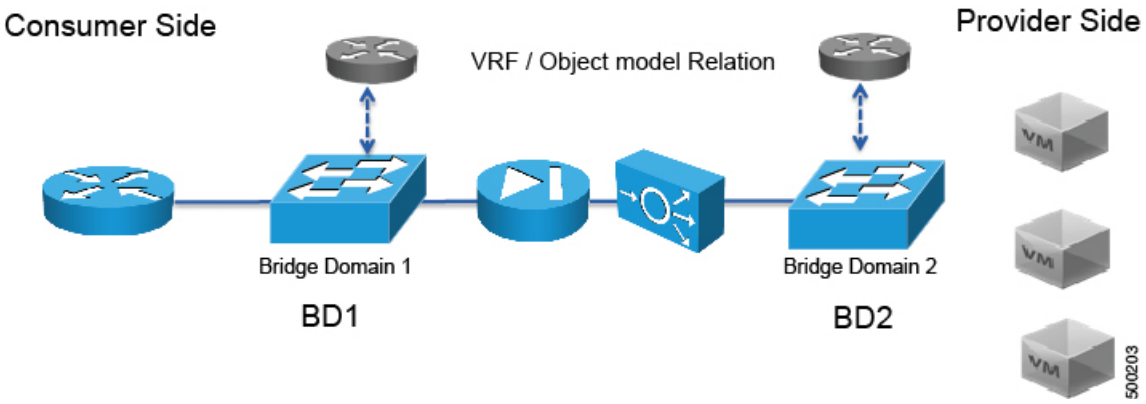
Figure 4: Example Service Graph Deployment



By using a service graph, you can install a service, such as an ASA firewall, once and deploy it multiple times in different logical topologies. Each time the graph is deployed, ACI takes care of changing the configuration on the firewall to enable the forwarding in the new logical topology.

Deploying a service graph requires bridge domains and VRFs, as shown in the following figure:

Figure 5: Bridge Domains and VRFs of a Service Graph



Note If you have some of the legs of a service graph that are attached to endpoint groups in other tenants, when you use the **Remove Related Objects of Graph Template** function in the GUI, the APIC does not remove contracts that were imported from tenants other than where the service graph is located. The APIC also does not clean endpoint group contracts that are located in a different tenant than the service graph. You must manually remove these objects that are in different tenants.

About Function Nodes

A function node represents a single service function. A function node has function node connectors, which represent the network requirement of a service function.

A function node within a service graph can require one or more parameters. The parameters can be specified by an endpoint group (EPG), an application profile, or a tenant VRF. Parameters can also be assigned at the time that you define a service graph. The parameter values can be locked to prevent any additional changes.

Note For Multi-Site configuration, up to 2 nodes can be deployed in a service graph. For non-Multi-Site configuration, up to 3 nodes can be deployed in one service graph.

About Function Node Connectors

A function node connector connects a function node to the service graph and is associated with the appropriate bridge domain and connections based on the graph's connector's subset. Each connector is associated with a VLAN or Virtual Extensible LAN (VXLAN). Each side of a connector is treated as an endpoint group (EPG), and whitelists are downloaded to the switch to enable communication between the two function nodes.

About Service Graph Connections

A service graph connection connects one function node to another function node.

About Terminal Nodes

Terminal nodes connect a service graph with the contracts. You can insert a service graph for the traffic between two application endpoint groups (EPGs) by connecting the terminal node to a contract. Once connected, traffic between the consumer EPG and provider EPG of the contract is redirected to the service graph.

Guidelines and Limitations for Configuring Service Graph

The following are guidelines and limitations for configuring Service Graph.

- A service-graph related configuration such as
  - A bridge domain (if used with a service graph) and service graph template should not contain the string “C-“ as part of its name.
  - A logical device should not contain the string “N-“ as part of its name.

About Service Graph Template Configuration Parameters

A service graph template can have configuration parameters, which are specified by the device package. Configuration parameters can also be specified by an EPG, application profile, or tenant context. A function node within a service graph template can require one or more configuration parameters. The parameter values can be locked to prevent any additional changes.

When you configure a service graph template and specify the values of the configuration parameters, the Application Policy Infrastructure Controller (APIC) passes the parameters to the device script that is within the device package. The device script converts the parameter data to the configuration that is downloaded onto the device.

Configuring a Service Graph Template Using the GUI

A service graph template is a sequence of Layer 4 to Layer 7 services functions, Layer 4 to Layer 7 services devices, or copy devices and their associated configuration, which can be provided by using function profiles. The service graph template must be associated with a contract to be "rendered"—or configured—on the Layer 4 to Layer 7 services device or copy device, and on the fabric.

Before you begin

You must have configured a tenant.

Step 1 On the menu bar, choose **Tenants** > **All Tenants**.

Step 2 In the Work pane, double-click the tenant's name.

Step 3 In the Navigation pane, choose **Tenant** ***tenant_name*** > **Services** > **L4-L7** > **Service Graph Templates**.

Step 4 In the Navigation pane, right-click **Service Graph Templates** and choose **Create a L4-L7 Service Graph Template**.

The **Create L4-L7 Service Graph Template** dialog box appears.

Step 5 If necessary, create one or more Layer 4 to Layer 7 services devices or copy devices:

a) Click the drop-down arrow in the **Device Clusters** pane of the **Create L4-L7 Service Graph Template** dialog box and choose **Create L4-L7 Devices** or **Create Copy Devices**.
The corresponding dialog box appears.

b) Follow the dialog box by entering the appropriate values displayed in the dialog box and clicking **Next** until finished.

Note For an explanation of a field in a dialog box, click the help icon in the top-right corner to display the help file.

c) When finished, click **Finish**.

You return to the **Create L4-L7 Service Graph Template** dialog box.

Step 6 Enter the appropriate values in the fields of the **Create L4-L7 Service Graph Template** dialog box.

Note For an explanation of a field in a dialog box, click the help icon in the top-right corner to display the help file.

Step 7 (Optional) (Only for cloning an existing service graph template) If you want to remove any of the nodes from the cloned service graph template, right-click a node that you want to remove and choose **Remove Node**.

Step 8 To create a service node, drag a Layer 4 to Layer 7 services device from the **Device Clusters** section and drop it between the consumer endpoint group and provider endpoint group. To create a copy node, drag and drop a copy device. This step is optional if you cloned an existing service graph template and the service graph template has all of the nodes that you want to use.

You can drag and drop multiple devices to create multiple nodes. The maximum number of service nodes is 3, although you can drag and drop greater numbers of other devices.

The location where you drop a copy device becomes the point in the data flow from where the copy device copies the traffic.

Step 9 If you created one or more service nodes, in the ***device_name*** **Information** section for each Layer 4 to Layer 7 services device, complete the fields. The fields vary depending on the device type.

Note For an explanation of a field, click the help icon in the top-right corner to display the help file.

Step 10 When finished, click **Submit**.

Step 11 (Optional) In the **Navigation** pane, click the service graph template.
The work pane displays a graphic topology of the service graph template.

Configuring a Service Graph Template Using the REST APIs

You can configure a service graph template using the following REST API:

```
<polUni>
  <fvTenant name="acme">
    <vnsAbsGraph name="G1">
      <vnsAbsTermNodeCon name="Input1">
        <vnsAbsTermConn name="C1">
            </vnsAbsTermConn>
      </vnsAbsTermNodeCon>
      <vnsAbsNode name="Node" funcType="GoTo">
        <vnsRsDefaultScopeToTerm
          tDn="uni/tn-acme/AbsGraph-G1/AbsTermNodeProv-Output1/outtmnl"/>
        <vnsAbsFuncConn name="inside">
          <vnsRsMConnAtt
            tDn="uni/infra/mDev-Insieme-Generic-1.0/mFunc-SubnetFunc/mConn-external"/>
        </vnsAbsFuncConn>
        <vnsAbsFuncConn name="outside">
          <vnsRsMConnAtt
            tDn="uni/infra/mDev-Insieme-Generic-1.0/mFunc-SubnetFunc/mConn-internal"/>
        </vnsAbsFuncConn>
        <vnsAbsDevCfg>
          <vnsAbsFolder key="oneFolder" name="f1">
            <vnsAbsParam key="oneParam" name="p1" value="v1"/>
          </vnsAbsFolder>
        </vnsAbsDevCfg>
        <vnsAbsFuncCfg>
          <vnsAbsFolder key="folder" name="folder1" devCtxLbl="C1">
            <vnsAbsParam key="param" name="param" value="value"/>
          </vnsAbsFolder>
          <vnsAbsFolder key="folder" name="folder2" devCtxLbl="C2">
            <vnsAbsParam key="param" name="param" value="value"/>
          </vnsAbsFolder>
        </vnsAbsFuncCfg>
        <vnsRsNodeToMFunc tDn="uni/infra/mDev-Insieme-Generic-1.0/mFunc-SubnetFunc"/>
      </vnsAbsNode>
      <vnsAbsTermNodeProv name="Output1">
        <vnsAbsTermConn name="C6">
            </vnsAbsTermConn>
      </vnsAbsTermNodeProv>
      <vnsAbsConnection name="CON1">
        <vnsRsAbsConnectionConns
          tDn="uni/tn-acme/AbsGraph-G1/AbsTermNodeCon-Input1/AbsTConn"/>
       <vnsRsAbsConnectionConns tDn="uni/tn-acme/AbsGraph-G1/AbsNode-Node/AbsFConn-inside"/>

      </vnsAbsConnection>
      <vnsAbsConnection name="CON3">
       <vnsRsAbsConnectionConns tDn="uni/tn-acme/AbsGraph-G1/AbsNode-Node/AbsFConn-outside"/>

        <vnsRsAbsConnectionConns
          tDn="uni/tn-acme/AbsGraph-G1/AbsTermNodeProv-Output1/AbsTConn"/>
      </vnsAbsConnection>
    </vnsAbsGraph>
  </fvTenant>
</polUni>
```

Applying a Service Graph Template to Endpoint Groups Using the GUI

The following procedure explains how to apply a service graph template to endpoint groups:

Before you begin

You must have created the following things:

- Application endpoint groups
- A service graph template

Step 1 On the menu bar, choose **Tenants** > **All Tenants**.

Step 2 In the Work pane, double click the tenant's name.

Step 3 In the Navigation pane, choose **Tenant** ***tenant_name*** > **Services** > **L4-L7** > **Service Graph Templates** > ***template_name*** .

Step 4 In the Navigation pane, right-click on the ***template_name*** that you want to apply to EPGs and choose **Apply L4-L7 Service Graph Template**.

The **Apply L4-L7 Service Graph Template To EPGs** dialog box appears. You will be associating a Layer 4 to Layer 7 service graph template to your consumer and provider endpoint groups.

Step 5 Configure a contract in the **Apply L4-L7 Service Graph Template To EPGS STEP 1> Contract** dialog box by entering the appropriate values:

a) If you are configuring an intra-EPG contract, place a check in the **Configure an Intra-EPG Contract** check-box and choose the EPG and network combination from the **EPG / Network** drop-down list.

b) If you are configuring a standard contract, choose the consumer/provider EPGs and network combinations in the appropriate drop-down lists.

c) Create a new contract or choose an existing one by clicking the appropriate radio button in the **Contract** field. If you select **Create A New Contract** and want to configure the filters for it, remove the check from the **No Filter (Allow All Traffic)** check-box. Click **+** to add filter entries and **Update** when complete.

Note For copy service graphs, contracts can only be used multiple times if they are applied to L3Out EPGs. Internal EPGs require an unshared contract.

Step 6 Click **Next**.

The **STEP 2 > Graph** dialog appears.

Step 7 In the ***device_name*** **Information** section, configure the required fields represented with a red box.

Note To include the connector in a preferred group (endpoint to endpoint communication without a contract), choose a configured policy from the **Service EPG Policy** drop-down list.

Step 8 Click **Next**.

The **STEP 3 > device_name Information** dialog appears.

Step 9 Configure parameters in **Required Parameters** and the **All Parameters** tab as required.

Step 10 Click **Finish**.

You now have an active service graph template.

Associating a Service Graph Template with a Contract Using the NX-OS-Style CLI

The following procedure associates a service graph template with a contract using the NX-OS-style CLI.

Step 1 Enter the configure mode.

Example:

```
apic1# configure
```

Step 2 Enter the configure mode for a tenant.

```
tenant tenant_name
```

Example:

```
apic1(config)# tenant t1
```

Step 3 Add a service graph.

```
l4l7 graph graph_name [contract contract_name]
```

Parameter	Description
graph	Name of the service graph.
contract	Name of the contract that is associated with this service graph instance. Specify the contract only if you want to create the service graph instance. You can simply configure a service graph (equivalent to the service graph template) without instantiating it.

Example:

```
apic1(config-tenant)# l4l7 graph G2 contract C2
```

Step 4 Add a node (service) in the service graph.

```
service node_name [device-cluster-tenant tenant_name] [device-cluster device_name] [mode
deployment_mode]
```

Parameter	Description
service	The name of the service node to add.
device-cluster-tenant	The tenant from which to import the device cluster. Specify this only if the device cluster is not in the same tenant in which the graph is being configured.
device-cluster	Name of the device cluster to use for this service node.

Parameter	Description
mode	The deployment mode. Possible values are: • `ADC_ONE_ARM`: Specifies one-arm mode. • `ADC_TWO_ARM`: Specifies two-arm mode. • `FW_ROUTED`: Specifies routed (GoTo) mode. • `FW_TRANS`: Specifies transparent (GoThrough) mode. • `OTHERS`: Specifies any other deployment mode. If the mode is not specified, then a deployment mode is not used.

Example:

The following example adds node `N1` to the device cluster `D4`, which is from tenant `t1`:

```
apic1(config-graph)# service N1 device-cluster-tenant t1 device-cluster D4
```

The following example adds node `N1` to the device cluster `D4`, which is from tenant `t1`, and uses the routed deployment mode:

```
apic1(config-graph)# service N1 device-cluster-tenant t1 device-cluster D4 mode FW_ROUTED
```

Step 5 Add the consumer connector.

```
connector connector_type [cluster-interface interface_type]
```

Parameter	Description
connector	The type of the connector in the service graph. Possible values are: • provider • consumer
cluster-interface	The type of the device cluster interface. Possible values are: • provider • consumer Do not specify this parameter if you are a service graph template in tenant `Common`.

Example:

```
apic1(config-service)# connector consumer cluster-interface consumer
```

Step 6 If the service interface is in a bridge domain, perform the following substeps:

a) Configure the bridge domain for the connectors by specifying the bridge domain information and tenant where the bridge domain is present.

```
bridge-domain tenant tenant_name name bridge_domain_name
```

Parameter	Description
tenant	Tenant that owns the bridge domain. You can only specify a bridge domain from same tenant or tenant `Common`. For example if you are in tenant `t1`, then you cannot specify the bridge domain from tenant `t2`.
name	Name of the bridge domain.

Example:

```
apic1(config-connector)# bridge-domain tenant t1 name bd2
```

b) Configure the direct server return (DSR) virtual IP address (VIP) for the connector.

```
dsr-vip ip_address
```

If you specify the DSR VIP, the Application Policy Infrastructure Controller (APIC) does not learn the VIP.

Parameter	Description
dsr-vip	The virtual IP address of the DSR for the connector.

Example:

```
apic1(config-connector)# dsr-vip 192.168.10.100
```

Step 7 If the service interface is in an L3Out, perform the following substeps:

a) Associate a tenant with the connector and then exit the connector configuration mode.

```
l4l7-peer tenant tenant_name out L3OutExternal epg epg_name
  redistribute redistribute_property
exit
```

Parameter	Description
tenant	The name of the tenant to associate with the connector.
out	The name of the Layer 3 outside.
epg	The name of the endpoint group.
redistribute	The properties of the redistribute protocol.

Example:

```
apic1(config-connector)# l4l7-peer tenant t1 out L3OutExternal epg L3ExtNet
  redistribute connected,ospf
apic1(config-connector)# exit
```

b) Repeat steps 5 and 7a for the provider.

Example:

```
apic1(config-service)# connector provider cluster-interface provider
apic1(config-connector)# l4l7-peer tenant t1 out L3OutInternal epg L3IntNet
  redistribute connected,ospf
apic1(config-connector)# exit
```

c) (Optional) Add a router and then exit the node configuration mode.

```
rtr-cfg router_ID
exit
```

Parameter	Description
rtr-cfg	The ID of the router.

Skip this step if you are creating a service graph template in tenant `Common`.

Example:

```
apic1(config-service)# rtr-cfg router-id1
apic1(config-service)# exit
```

Step 8 Configure connections for the consumer and provider and exit the service graph configuration mode.

```
connection connection_name {terminal terminal_type service node_name connector connector_type} |
  {intra_service service1 node_name connector1 connector_type service2 node_name connector2
connector_type}
exit
```

Parameter	Description
connection	The name of the connection.
terminal	Connects a service node to the terminal. Specifies the type of the terminal. Possible values are: • provider • consumer
service service1 service2	The name of the service node to add. service is used only with terminal. service1 and service2 are used only with intra_service.
connector connector1 connector2	The type of the connector. Possible values are: • provider • consumer connector is used only with terminal. connector1 and connector2 are used only with intra_service.
intra_service	Connects a service node to another node.

Example:

The following example configures the connections of a single node graph:

```
apic1(config-graph)# connection CON1 terminal consumer service N1 connector consumer
apic1(config-graph)# connection CON2 terminal provider service N2 connector provider
apic1(config-graph)# exit
```

The following example configures the connections of a two node graph:

```
apic1(config-graph)# connection CON1 terminal consumer service N1 connector consumer
apic1(config-graph)# connection CON2 intra_service service1 N1 connector1 provider service2 N2
connector2 consumer
apic1(config-graph)# connection CON3 terminal provider service N2 connector provider
apic1(config-graph)# exit
```

Step 9 Exit the configuration mode.

Example:

```
apic1(config-tenant)# exit
apic1(config)# exit
```

Creating a Function Profile Using the GUI

A function profile provides the default values for your service graph template. The following procedure explains how to create a new function profile.

Step 1 On the menu bar, choose **Tenants** > **All Tenants**.

Step 2 In the Work pane, double click the tenant's name.

Step 3 In the Navigation pane, choose **Tenant** *tenant_name* > **Services** > **L4-L7** > **Function Profiles**.

Step 4 Right-click **Function Profiles** and choose **Create L4-L7 Services Function Profile**.

Step 5 In the **Create L4-L7 Services Function Profile** dialog box, enter the appropriate values in the fields as required, except as specified below:

a) In the **Profile Group** drop-down list, choose **Create Function Profile Group**.

A profile group is a mechanism that allows you to group your profiles together for organizational purposes. For example, you may want to create a profile for your Web, legacy, or e-mail applications. You can create groups and then you can put your profiles into those groups. You may see that you already have an existing group available, but if you do not, then you can create a new one by naming it and providing a description in the **Create L4-L7 Services Function Profile Group** window.

Step 6 In the **Create L4-L7 Services Function Profile Group** dialog box, enter the appropriate values in the fields as required

Step 7 Click **Submit**.

You return to the **Create L4-L7 Services Function Profile** dialog box with a successfully completed and saved a profile group, which now appears in the **Create L4-L7 Services Function Profile** dialog box.

A profile is created for a particular service function. What you choose from the **Device Function** drop-down list in the **Create L4-L7 Services Function Profile** is the function for which you are writing a profile. From the drop-down list, you will see a list of device packages with service functions available in the Application Policy Infrastructure Controller (APIC) after you have imported the device packages.

Step 8 In the **Create L4-L7 Services Function Profile** dialog box, remove the check from the **Copy Existing Profile Parameters** check box.

Step 9 In the **Device Function** drop-down list, choose a device function.
Options are displayed with the various parameters that are part of that function. The purpose of the profile is to provide the default values for the parameters.

Note At this point, the parameters do not have any values until you add them. The values you add are then used as the default values. The function profiles can be used by the graph templates after you provide these values. These values are applied to the graph template as default values, which means that if you use the graph templates and you do not provide a value for that particular parameter, then the APIC looks up the profile and see if the value is there. If it is there, then the APIC uses that.

Step 10 Add values in the **Features and Parameters** section at the bottom of the **Create L4-L7 Services Function Profile** dialog box. There are two tabs, **Basic Parameters** and **All Parameters**. The **Basic Parameters** tab includes a list of parameters that are marked as mandatory (required) in the package. The **All Parameters** tab includes a list of the required parameters as well as some additional/optional parameters for advanced configurations. We expose the **Basic Parameters** because they are part of the basic configuration and the administrator is expected to fill these out. **All Parameters** are optional, so unless you want to customize the functionality, these parameters can be left out.

Step 11 (Optional) Create a cloud orchestrator mode function profile as follows:

a) Double-click on a folder or parameter in the **All Parameters** or **Basic Parameters** tab.
The row that corresponds to the chosen folder or parameter opens.

b) Specify the **Path from Schema**:

- If specifying a path for a folder, the **Path from Schema** column lists all the possible folder paths in a drop-down list. Choose the path that the folder maps to in the schema.
- If specifying a path for a parameter:

1. click the edit icon in the **Path from Schema** field. The **Manage Path-From-Schema** dialog appears.
2. Click to enable **Specify Path-From-Schema**.
3. Click the **Path** drop-down arrow, and choose a path.
4. Click the **+** in the parameter editor and choose a parameter from the drop-down list.
5. When finished, click **Ok**. You return to the **Create L4-L7 Services Function Profile**.
6. (Optional) Enter values in the following fields:
   - **Value**–Enter a value in the if you want UI to show a default value while deploying the graph for the chosen parameter.
   - **Hint**–Specify text that displays when a value is entered in the UI for the chosen parameter while deploying the graph.

c) Click **Update**.

Step 12 Click **Submit**.
Now you have completed and saved your function profile.

Using an Existing Function Profile to Create a New Function Profile Using the GUI

This procedure uses an existing function profile to create a new function profile.

Step 1 On the menu bar, choose **Tenants** > **All Tenants**.

Step 2 In the Work pane, double click the tenant's name.

Step 3 In the Navigation pane, choose **Tenant** ***tenant_name*** > **Services** > **L4-L7** > **Function Profiles**.

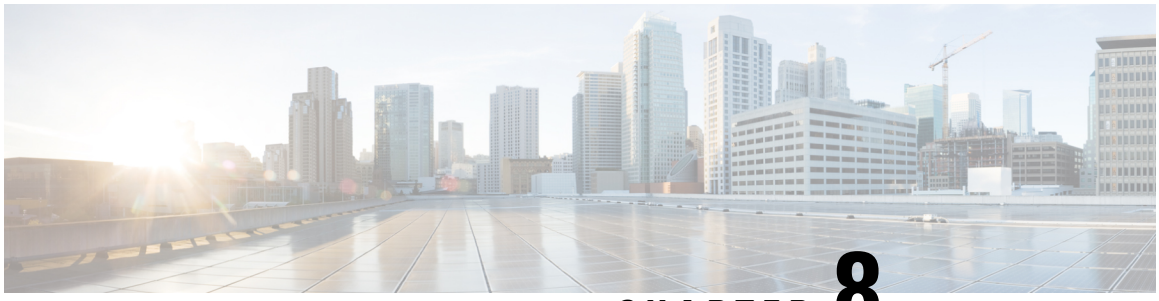
Step 4 Right click **Function Profiles** and choose **Create L4-L7 Services Function Profile**.

Step 5 In the **Create L4-L7 Services Function Profile** dialog box, fill in the fields as required, except as specified below:

a) In the **Profile** drop-down list, choose an existing profile that is supplied by the vendor.
The parameters are populated for your new profile based on the profile that you chose.

b) Change or add parameters to this existing profile as necessary.

Step 6 Click **Submit**.



CHAPTER 8

Configuring Route Peering

- About Route Peering, on page 49
- Open Shortest Path First Policies, on page 50
- Border Gateway Protocol Policies, on page 54
- Selecting an L3extOut Policy for a Cluster, on page 57
- Route Peering End-to-End Flow, on page 58
- Cisco Application Centric Infrastructure Fabric Serving As a Transit Routing Domain, on page 60
- Configuring Route Peering Using the GUI, on page 61
- Configuring Route Peering Using the NX-OS-Style CLI, on page 65
- Troubleshooting Route Peering, on page 67

About Route Peering

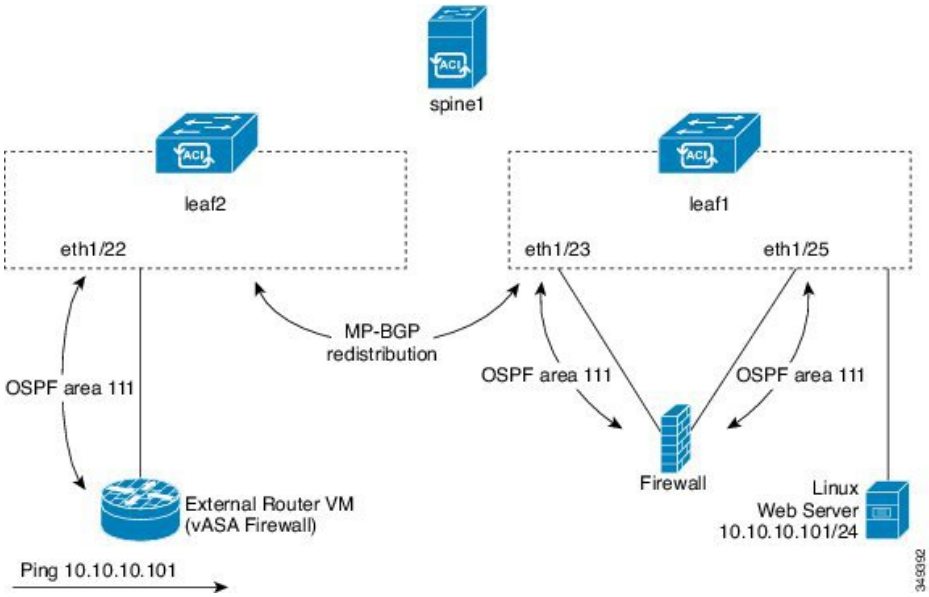
Route peering is a special case of the more generic Cisco Application Centric Infrastructure (ACI) fabric as a transit use case, in which route peering enables the ACI fabric to serve as a transit domain for Open Shortest Path First (OSPF) or Border Gateway Protocol (BGP) protocols. A common use case for route peering is route health injection, in which the server load balancing virtual IP is advertised over OSPF or internal BGP (iBGP) to clients that are outside of the ACI fabric. You can use route peering to configure OSPF or BGP peering on a service device so that the device can peer and exchange routes with the ACI leaf switch to which it is connected.

The following protocols are supported for route peering:

- OSPF
- OSPFv3
- iBGPv4
- iBGPv6
- Static routes

The following figure shows how route peering is commonly deployed:

Figure 6: Common Route Peering Topology



As shown in the figure, a Web server's public IP address is advertised to an external router through a firewall by deploying a service graph with route peering configured. You must deploy OSPF routing policies on each leg of the firewall. This is typically done by deploying `l3extOut` policies. This enables the Web server reachability information to be advertised over OSPF through the firewall to the border leaf switch and to the external router.

Route distribution between leaf switches in the fabric is internally accomplished over Multi-Protocol Border Gateway Protocol (MP-BGP).

For a more detailed example of the route peering topology, see Route Peering End-to-End Flow, on page 58.

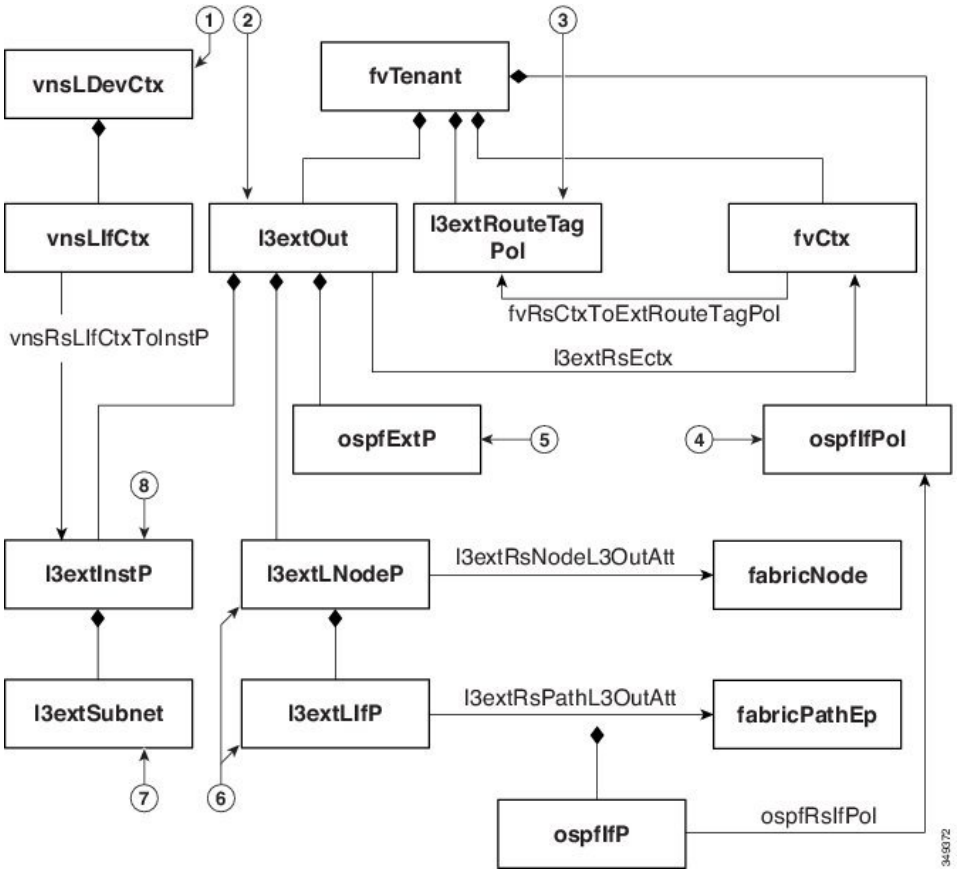
For more information about configuring `l3extOut` policies, see the *Cisco Application Centric Infrastructure Fundamentals Guide*.

Note Point-to-point non-broadcast mode is not supported on an Adaptive Security Appliance (ASA). You must remove the point-to-point non-broadcast mode configuration from the Application Policy Infrastructure Controller (APIC) if the configuration exists.

Open Shortest Path First Policies

To configure route peering, you must first create one or more `l3extOut` policies and deploy them on the fabric leaf nodes where the service device is connected. These `l3extOut` policies specify the Open Shortest Path First (OSPF) parameters that you must enable on the fabric leaf. The policies are very similar to the `l3extOut` policies that are used for external communication. The following figure illustrates the route peering object relations.

Figure 7: OSPF Route Peering Object Relations



1. `vnsLDevCtx`—Device selection policy.
2. `l3extOut`—Contains all OSPF policies for a single area.
3. `l3extRouteTagPol`—Every context used by route peering needs a unique route tag to avoid OSPF loops. The OSPF routes that are learned from one leg will not be learned on the other leg unless the route tags are different.
4. `ospfIfPol`—OSPF per interface policy.
5. `ospfExtP`—OSPF per area policy.
6. `l3extLNodeP/l3extLIfP`—The nodes or ports on which this `l3extOut` is deployed.
7. `l3extSubnet`—Subnets to export from or import into the fabric.
8. `l3extInstP`—Prefix-based EPG.

Two example `l3extOut` policies, `OspfExternal` and `OspfInternal`, are shown below. These policies are deployed on the external and internal legs of the firewall device in Figure 6: Common Route Peering Topology, on page 50. The `l3extOut` policy specifies one or more prefix-based EPGs (`l3extInstP`), which control how traffic is classified by the fabric leaf and also how routes are imported from and exported to the service device. The `l3extOut` policy contains the OSPF per-area policy (`ospfExtP`) and one or more OSPF interface policies (`ospfIfPol`) that are specified under it.

The following example shows an OSPF area with area-Id being configured with a value of "100":

```
<ospfExtP areaId="100" areaType="regular" areaCtrl="redistribute"/>
```

The area type is set to "regular" and the area control attribute is set to "redistribute".

The OSPF interface policy specifies one or more OSPF interface timers:

```
<ospfIfPol name="ospfIfPol" ctrl="mtu-ignore" nwT="bcast" xmitDelay="1" helloIntvl="10"
  deadIntvl="40" status="created,modified"/>
```

If default timers are fine, then you do not need to specify this policy. This policy allows certain timers to be modified from default values and is associated with one or more interfaces by using the following relation:

```
<l3extRsPathL3OutAtt tDn="topology/pod-l/paths-101/pathep-[eth1/25]" ifInstT="ext-svi"
  encap="vlan-3844" addr="30.30.30.100/28" mtu="1500"/>
```

The attributes of the `l3extRsPathL3OutAtt` relation are as follows:

- `ifInstT`—The logical interface type, which is typically "ext-svi".
- `encap`—You must specify a VLAN encapsulation when creating this interface. The encapsulation is pushed to the service device.
- `addr`— The IP address of the SVI interface that was created on the fabric leaf where this `l3extOut` is deployed.

The following policy controls where the `l3extOut` policy is deployed:

```
<l3extNodeP name="bLeaf-101">
  <l3extRsNodeL30utAtt tDn="topology/pod-l/node-101" rtrId="180.0.0.11"/>
  <l3extLIfP name="portlf">
    <l3extRsPathL30utAtt tDn="topology/pod-1/paths-101/pathep-teth1/251"
      ifInstT="ext-svi" encap="vlan-3844" addr="30.30.30.100/28" mtu="1500"/>
    <ospfIfP authKey="tecom" authType="md5" authKeyId='1'>
      <ospfRslfPol tnOspfIfPolName="ospfIfPol"/>
    </ospflfP>
  </l3extLIfP>
</l3extLNodeP>
```

The `l3extOut` policy must be deployed to the same leaf ports to which the service device is connected.

The `scope=import-security` attribute does the following things:

- Controls the flow of traffic in the data plane
- Acts as a directive to the external device to advertise this route

Note For route peering to work correctly, the `l3extRsPathL3OutAtt` relation must point to the same fabric destination as the `RsCIfPathAtt` relation under the `vnsCDev` that represents the device.

`OspfExternal` Policy

`OspfInternal` Policy

Virtual Services

```
<polUni>
  <fvTenant name="common">
    <fvCtx name="commonctx">
      <fvRsCtxToExtRouteTagPol tnL3extRouteTagPolName="myTagPol"/>
    </fvCtx>
    <l3extRouteTagPol tag="212" name="myTagPol"/>
    <l3extOut name="OspfExternal" status="created,modified">
      <l3extLNodeP name="bLeaf-101">
        <l3extRsNodeL3OutAtt tDn="topology/pod-1/node-101" rtrId="180.0.0.8/28"/>
        <l3extLIfP name="portIf">
          <l3extRsPathL3OutAtt tDn="topology/pod-1/paths-101/pathep-[eth1/23]"
            ifInstT="ext-svi" encap="vlan-3843" addr="40.40.40.100/28" mtu="1500"/>
          <ospfIfP authKey="tecom" authType="md5" authKeyId='1'>
            <ospfRsIfPol tnOspfIfPolName="ospfIfPol"/>
          </ospfIfP>
        </l3extLIfP>
      </l3extLNodeP>
      <ospfExtP areaId="100" areaType="regular" areaCtrl="redistribute"/>
      <l3extInstP name="ExtInstP">
        <l3extSubnet ip="40.40.40.100/28" scope="import-security"/>
        <l3extSubnet ip="10.10.10.0/24" scope="import-security"/>
      </l3extInstP>
      <l3extRsEctx tnFvCtxName="commonctx"/>
    </l3extOut>
   <ospfIfPol name="ospfIfPol" ctrl="mtu-ignore" nwT="bcast" xmitDelay="1" helloIntvl="10"

      deadIntvl="40" status="created,modified"/>
  </fvTenant>
</polUni>
```

```
<polUni>
  <fvTenant name="tenant1">
    <l3extRouteTagPol tag="213" name="myTagPol"/>
    <fvCtx name="tenant1ctx1">
      <fvRsCtxToExtRouteTagPol tnL3extRouteTagPolName="myTagPol"/>
    </fvCtx>
    <l3extOut name="OspfInternal" status="created,modified">
      <l3extLNodeP name="bLeaf-101">
        <l3extRsNodeL3OutAtt tDn="topology/pod-1/node-101" rtrId="180.0.0.11"/>
        <l3extLIfP name="portIf">
          <l3extRsPathL3OutAtt tDn="topology/pod-1/paths-101/pathep-[eth1/25]"
            ifInstT="ext-svi" encap="vlan-3844" addr="30.30.30.100/28" mtu="1500"/>
          <ospfIfP authKey="tecom" authType="md5" authKeyId='1'>
            <ospfRsIfPol tnOspfIfPolName="ospfIfPol"/>
          </ospfIfP>
        </l3extLIfP>
      </l3extLNodeP>
      <ospfExtP areaId="100" areaType="regular" areaCtrl="redistribute"/>
      <l3extInstP name="IntInstP">
        <l3extSubnet ip="30.30.30.100/28" scope="import-security"/>
        <l3extSubnet ip="20.20.20.0/24" scope="import-security"/>
      </l3extInstP>
      <l3extRsEctx tnFvCtxName="tenant1ctx1"/>
    </l3extOut>
   <ospfIfPol name="ospfIfPol" ctrl="mtu-ignore" nwT="bcast" xmitDelay="1" helloIntvl="10"
```

```
      deadIntvl="40" status="created,modified"/>
  </fvTenant>
</polUni>
```

The `OspfExternalInstP` policy specifies that prefixes 40.40.40.100/28 and 10.10.10.0/24 must be used for prefix-based endpoint association. The policy also instructs the fabric to export prefix 20.20.20.0/24 to the service device.

```
<l3extInstP name="OspfExternalInstP">
  <l3extSubnet ip="40.40.40.100/28" scope="import-security"/>
  <l3extSubnet ip="10.10.10.0/24" scope="import-security"/>
  <l3extSubnet ip="20.20.20.0/24" scope="export"/>
</l3extInstP>
```

The `bleaf-101` policy controls where this `l3extOut` policy is deployed.

```
<l3extLNodeP name="bLeaf-101">
  <l3extRsNodeL3OutAtt tDn="topology/pod-1/node-101" rtrId="180.0.0.8/28"/>
  <l3extLIfP name="portIf">
    <l3extRsPathL3OutAtt tDn="topology/pod-1/paths-101/pathep-[eth1/23]"
      ifInstT="ext-svi" encap="vlan-3843" addr="40.40.40.100/28" mtu="1500"/>
    <!-- <ospfIfP authKey="tecom" authType="md5" authKeyId='1'> -->
    <ospfIfP>
      <ospfRsIfPol tnOspfIfPolName="ospfIfPol"/>
    </ospfIfP>
  </l3extLIfP>
</l3extLNodeP>
```

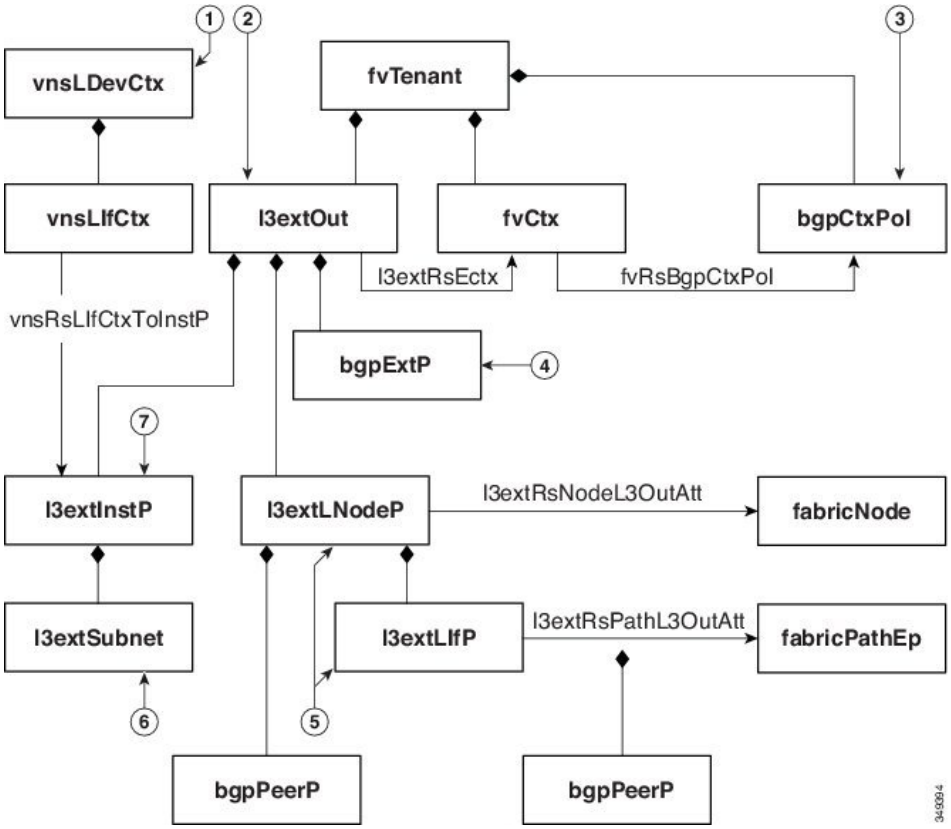
You can deploy virtual services with route peering, although the `l3extRsPathL3OutAtt` validation with the `vnsCIf` object is not performed. The datapath will work only if the `l3extOut` object is deployed to the correct leaf to which the virtual service device is connected.

Border Gateway Protocol Policies

You can configure route peering using internal Border Gateway Protocol (iBGP) on the device's external interface and static routes on the internal interface. You cannot configure iBGP on both the internal and external interfaces of the device without extra configuration, as the interfaces must be in different autonomous systems and inter-autonomous system redistribute policies do not get pushed down.

The following figure illustrates the route peering object relations:

Figure 8: iBGP Route Peering Object Relations



1. `vnsLDevCtx`—Device selection policy.
2. `l3extOut`—Contains all BGP policies for a single autonomous system.
3. `bgpCtxPol`—Per-context BGP timers.
4. `bgpExtP`—BGP per ASN policy.
5. `l3extLIfP/l3extLNodeP`—Controls to which nodes or ports these endpoint groups (EPGs) are deployed.
6. `l3extSubnet`—Subnets to export from and import into the fabric.
7. `l3extInstP`—Prefix-based EPG.

The following policy configures iBGPv4/v6 on the external interface:

```
<polUni>
  <fvTenant name="common">
    <fvCtx name="commonctx">
      <fvRsBgpCtxPol tnBgpCtxPolName="timer-3-9"/>
      <fvRsCtxToExtRouteTagPol tnL3extRouteTagPolName="myTagPol"/>
    </fvCtx>
    <l3extRouteTagPol tag="212" name="myTagPol"/>
   <bgpCtxPol grCtrl="helper" holdIntvl="9" kaIntvl="3" name="timer-3-9" staleIntvl="30"/>

    <l3extOut name="BgpExternal" status="created,modified">
      <l3extLNodeP name="bLeaf-101">
        <!-- <bgpPeerP addr="40.40.40.102/32 "ctrl="send-com"/> -->
        <l3extRsNodeL3OutAtt tDn="topology/pod-1/node-101" rtrId="180.0.0.8/28>"
```

```
          <l3extLoopBackIfP addr="50.50.50.100/32"/>
        </l3extRsNodeL3OutAtt>
        <l3extLIfP name="portIf">
          <l3extRsPathL3OutAtt tDn="topology/pod-1/paths-101/pathep-[eth1/23]"
            ifInstT="ext-svi" encap="vlan-3843" addr="40.40.40.100/28 "mtu="1500">
            <bgpPeerP addr="40.40.40.102/32 "ctrl="send-com"/>
          </l3extRsPathL3OutAtt>
        </l3extLIfP>
      </l3extLNodeP>
      <bgpExtP/>
      <l3extInstP name="ExtInstP">
        <l3extSubnet ip="40.40.40.100/28 "scope="import-security"/>
        <l3extSubnet ip="10.10.10.0/24 "scope="import-security"/>
        <l3extSubnet ip="20.20.20.0/24 "scope="export-rtctrl"/>
      </l3extInstP>
      <l3extRsEctx tnFvCtxName="commonctx"/>
    </l3extOut>
  </fvTenant>
</polUni>
```

iBGP peers can be configured at the physical interface level or the loopback level. The following example shows a iBGP peer configured at the physical interface level:

```
<l3extLIfP name="portIf">
          <l3extRsPathL3OutAtt tDn="topology/pod-1/paths-101/pathep-[eth1/23]"
            ifInstT="ext-svi" encap="vlan-3843" addr="40.40.40.100/28 "mtu="1500">
            <bgpPeerP addr="40.40.40.102/32 "ctrl="send-com"/>
          </l3extRsPathL3OutAtt>
</l3extLIfP>
```

In this case, the iBGP process that is running on the fabric uses the switch virtual interface (SVI) IP address 40.40.40.100/28 to peer with its neighbor. The neighbor is the service device at IP address 40.40.40.102/32.

In the following example, the iBGP peer definition has been moved to the logical node level (under `l3extLNodeP`) and a loopback interface has been configured:

```
      <l3extLNodeP name="bLeaf-101">
        <bgpPeerP addr="40.40.40.102/32 "ctrl="send-com"/>
        <l3extRsNodeL3OutAtt tDn="topology/pod-1/node-101" rtrId="180.0.0.8/28>"
          <l3extLoopBackIfP addr="50.50.50.100/32"/>
        </l3extRsNodeL3OutAtt>
        <l3extLIfP name="portIf">
          <l3extRsPathL3OutAtt tDn="topology/pod-1/paths-101/pathep-[eth1/23]"
            ifInstT="ext-svi" encap="vlan-3843" addr="40.40.40.100/28 "mtu="1500">
          </l3extRsPathL3OutAtt>
        </l3extLIfP>
      </l3extLNodeP>
```

In this case, the iBGP process uses the loopback address to peer with its neighbor. If no loopback is configured, then the fabric uses the IP address that is specified by rtrId to peer with the neighbor.

In such cases, the device needs a route to reach the SVI. This is typically configured using graph parameters, as shown by the following example for ASA, where IP address 50.50.50.0 is reachable from IP address 40.40.40.100:

```
<vnsAbsFolder name="ExtRouteCfg" key="StaticRoute">
              <vnsAbsFolder name="route1" key="route">
                <vnsAbsParam name="network" key="network" value="50.50.50.0"/>
                <vnsAbsParam name="netmask" key="netmask" value="255.255.255.0"/>
                <vnsAbsParam name="gateway" key="gateway" value="40.40.40.100"/>
              </vnsAbsFolder>
              <vnsAbsFolder name="route2" key="ipv6_route">
                <vnsAbsParam name="prefix" key="prefix" value="2005::/64"/>
                <vnsAbsParam name="gateway" key="gateway" value="2004::2828:2866"/>
```

```
            </vnsAbsFolder>
</vnsAbsFolder>
```

The following example shows static route configuration on the fabric for the internal interface of the device:

```
<polUni>
  <fvTenant name="tenant11">
    <l3extOut name="StaticInternal" status="created,modified">
      <l3extLNodeP name="bLeaf-201">
        <l3extRsNodeL3OutAtt tDn="topology/pod-1/node-101" rtrId="180.0.0.11>"
          <ipRouteP ip="20.20.20.0/24>"
            <ipNexthopP nhAddr="30.30.30.102/32"/>
          </ipRouteP>
        </l3extRsNodeL3OutAtt>
        <l3extLIfP name="portIf">
          <l3extRsPathL3OutAtt tDn="topology/pod-1/paths-101/pathep-[eth1/25]"
            ifInstT="ext-svi" encap="vlan-3844" addr="30.30.30.100/28 "mtu="1500"/>
        </l3extLIfP>
      </l3extLNodeP>
      <l3extInstP name="IntInstP">
        <l3extSubnet ip="20.20.20.0/24 "scope="import-security"/>
      </l3extInstP>
      <l3extRsEctx tnFvCtxName="tenant1ctx1"/>
    </l3extOut>
  </fvTenant>
</polUni>
```

Selecting an L3extOut Policy for a Cluster

A specific `l3extOut` policy can be associated with a logical device's interface using its selection policy `vnsLIfCtx`. The following example shows how this is achieved:

```
<vnsLDevCtx ctrctNameOrLbl="webCtrct1" graphNameOrLbl="WebGraph" nodeNameOrLbl="FW">
    <vnsRsLDevCtxToLDev tDn="uni/tn-tenant1/lDevVip-Firewall"/>
    <vnsRsLDevCtxToRtrCfg tnVnsRtrCfgName="FwRtrCfg"/>
    <vnsLIfCtx connNameOrLbl="internal">
        <vnsRsLIfCtxToInstP tDn="uni/tn-tenant1/out-OspfInternal/instP-IntInstP"
          status="created,modified"/>
        <vnsRsLIfCtxToLIf tDn="uni/tn-tenant1/lDevVip-Firewall/lIf-internal"/>
    </vnsLIfCtx>
    <vnsLIfCtx connNameOrLbl="external">
        <vnsRsLIfCtxToInstP tDn="uni/tn-common/out-OspfExternal/instP-ExtInstP"
          status="created,modified"/>
        <vnsRsLIfCtxToLIf tDn="uni/tn-tenant1/lDevVip-Firewall/lIf-external"/>
    </vnsLIfCtx>
</vnsLDevCtx>
```

The `vnsRsLIfCtxToInstP` relation is used to select a particular prefix-based EPG that (`l3extInstP`) is associated with this leg of the service device. You can specify the `redistribute` protocol redistribute property on this relation. The default value for the `redistribute` property is "ospf,bgp". Leaving `redistribute` at the default value causes the Application Policy Infrastructure Controller (APIC) to auto-detect the routing protocols that are configured on each leg and push the appropriate redistribute settings. The automatic settings always redistribute from an interior gateway protocol (OSPF) to an exterior gateway protocol (BGP).

If you want to use a specific redistribute setting, such as static or connected, then you can add those settings to this relation. For example, `redistribute="ospf,bgp,static"` causes the auto-detected settings and `redistribute-static` to be pushed to the service device.

Setting this property to a specific value that does not include the defaults, such as `redistribute="ospf,static,connected"`, causes those exact settings to be pushed to the service device. This is useful in scenarios in which you want to override the defaults that are chosen by the APIC.

Note The relation points to an EPG (`l3extInstP`) and not to the `l3extOut` itself, as there can be multiple such EPGs under an `l3extOut` policy, and different device selection policies could point to those EPGs. This allows for fine control of which prefixes are imported or exported by different service graphs.

The `vnsRsLDevCtxToRtrCfg` relation is used to select a particular `vnsRtrCfg` policy for this device selector. `vnsRtrCfg` policies are needed to specify the router ID that is used by routing protocols, such as Open Shortest Path First (OSPF) or internal Border Gateway Protocol (iBGP), and must be supplied by the user. This router ID is sent to the device.

The following code is an example `vnsRtrCfg` policy:

```
<vnsRtrCfg name="FwRtrCfg" rtrId="180.0.0.10”/>
```

The associated concrete device must have a `vnsRsCIfPathAtt` object, which deploys the device to the same fabric leaf as shown below:

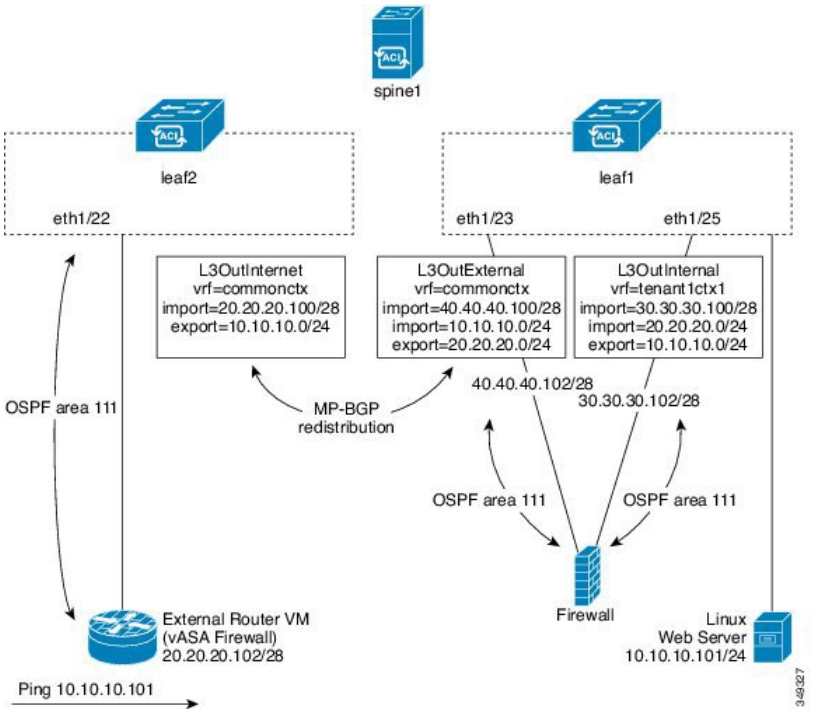
```
<vnsCDev name="ASA">
    <vnsCIf name="Gig0/0">
        <vnsRsCIfPathAtt tDn="topology/pod-1/paths-101/pathep-[eth1/23]"/>
    </vnsCIf>
    <vnsCIf name="Gig0/1">
        <vnsRsCIfPathAtt tDn="topology/pod-1/paths-101/pathep-[eth1/25]"/>
    </vnsCIf>
    <vnsCMgmt name="devMgmt" host="{{asaIp}}" port="443"/>
    <vnsCCred name="username" value="admin"/>
    <vnsCCredSecret name="password" value="insieme"/>
</vnsCDev>
```

Note When route peering is configured, you do not need to configure bridge domains on the `vnsLIfCtx` selectors. Attempting to configure both bridge domain relations (`vnsRsLIfCtxToBD`) and `l3extInstP` relations (`vnsRsLIfCtxToInstP`) will result in a fault.

Route Peering End-to-End Flow

The following figure shows how route peering works end-to-end.

Figure 9: Route Peering End-to-End Flow



The figure shows an example two leaf switch, single spine switch topology where a Linux web server's IP address is advertised to an external router using route peering. The Linux web server is at IP address 10.10.10.101/24 and is hosted on an ESX server that is connected to `leaf1`. A regular bridge domain-based endpoint group (EPG) is deployed to represent traffic that originates from the web server.

You deploy a service graph that comprises a two-arm routable firewall, with both arms being connected to `leaf1`. There is a virtual routing and forwarding (VRF)-split on the firewall device, meaning that each arm of the firewall is connected to the leaf switch in a different VRF (context). The VRF-split is necessary to ensure that traffic is routed through the service device, rather than being short-circuited by the leaf switch. The external traffic is represented by an `l3extOut` (`L3OutInternet`) that is deployed on `leaf2`. `leaf2` can be viewed as a fabric border-leaf switch in this scenario. You deploy a contract between `L3OutInternet` and the web server EPG. This contract is associated with a service graph that encompasses the firewall device.

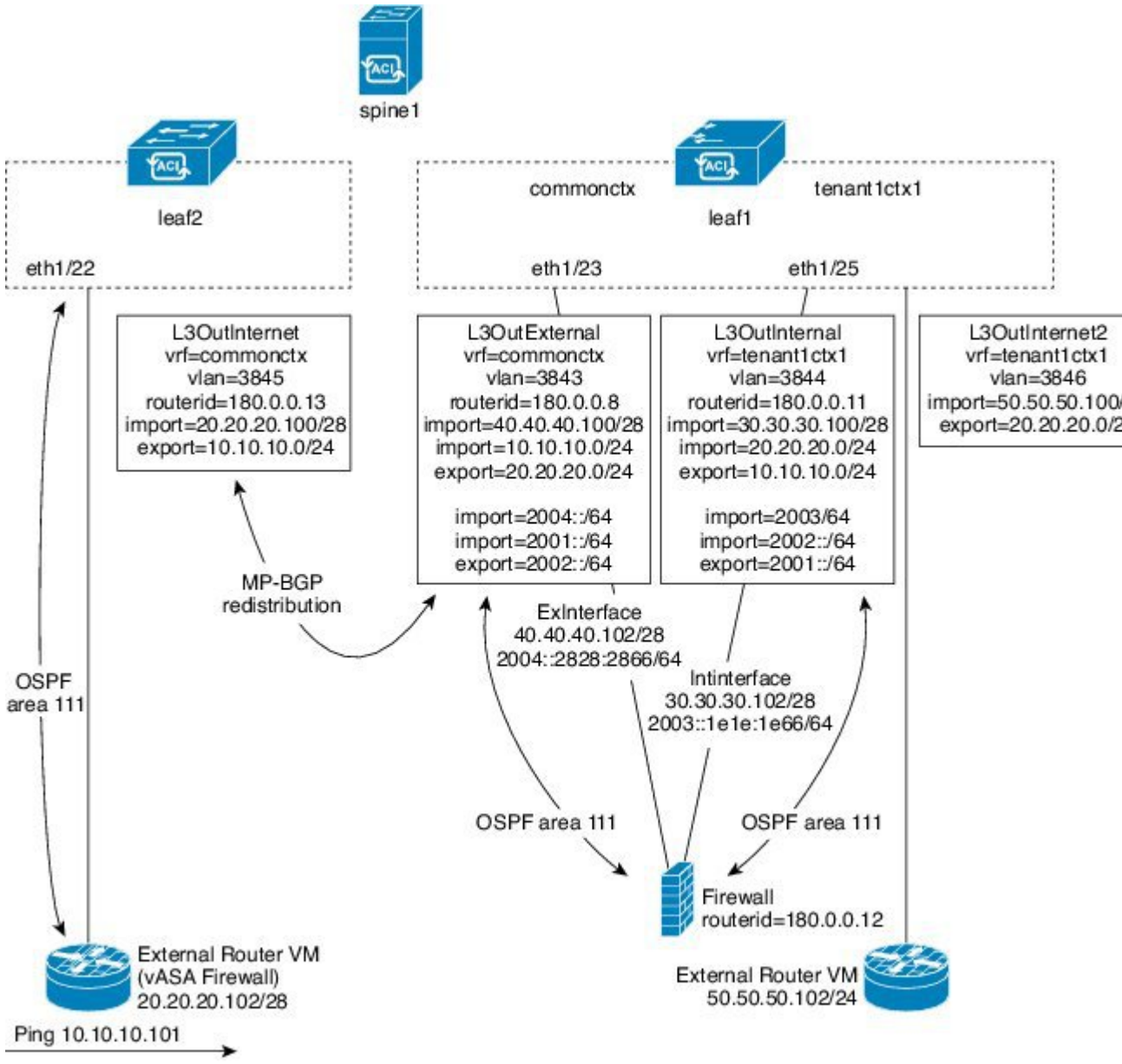
To publish the web server route to the external world, you deploy two `l3extOuts`—`L3OutExternal` and `L3OutInternal`—to the leaf switch ports to which the service device is connected. As a result, Open Shortest Path First (OSPF) peering sessions are established between the leaf switch and the firewall in both of the contexts (`commonctx` and `tenant1ctx1`). The export attribute on these `l3extOuts` control how the routing information is advertised to the border leaf switch. Routes are exchanged internally between the fabric leaf switches using Multiprotocol Border Gateway Protocol (MP-BGP) redistribution.

Ultimately, the web server route is advertised to the external router (IP address 20.20.20.102) using a separate OSPF session. This results in the external router being able to ping the web server without any manual static route configuration.

Cisco Application Centric Infrastructure Fabric Serving As a Transit Routing Domain

You can deploy the Cisco Application Centric Infrastructure (ACI) fabric as a transit routing domain, which is useful when the ACI point of delivery (POD) serves as a transit routing domain between other PODs. In the following illustration, two external `l3extOut`s—`L3OutInternet` and `L3OutInternet2`—are deployed on two border leaf switches. There is a contract associated between these `l3extOut`s, and the contract is attached to a single node service graph containing a firewall service device.

Figure 10: ACI Fabric Serving As a Transit Routing Domain



Two additional `l3extOut`s are deployed on the external and internal legs of the firewall device to establish Open Shortest Path First (OSPF) peering sessions between them. By appropriately configuring the import security control (the `import-security` attribute), you can control which routes are allowed to transit the ACI fabric to the border leaf switches.

Configuring Route Peering Using the GUI

You must perform the following tasks to configure route peering:

1. Create a static VLAN pool that will be used for the encapsulation VLAN between the device and the Cisco Application Centric Infrastructure (ACI) fabric.

   See Creating a Static VLAN Pool Using the GUI, on page 61.

2. Create an external routed domain that will tie together the location (leaf node/path) of the device and the VLAN pool.

   See Creating an External Routed Domain Using the GUI, on page 62.

3. Create an external routed network, which is used to specify the routing configuration in the ACI fabric for route peering.

   See Creating an External Routed Network Using the GUI, on page 62.

4. Create a new router configuration to specify the router ID that will be used on the device.

   See Creating a Router Configuration Using the GUI, on page 64.

5. Create a service graph association, which involves associating the external routed network policy and router configuration with a device selection policy.

   See Creating a Service Graph Association Using the GUI, on page 64.

Creating a Static VLAN Pool Using the GUI

Before creating an external routed network configuration, you must create a static VLAN pool that will be used for the encapsulation VLAN between the device and the fabric.

Step 1 On the menu bar, choose **Fabric** > **Access Policies**.

Step 2 In the Navigation pane, choose **Pools** > **VLAN**.

Step 3 In the Work pane, choose **Actions** > **Create VLAN Pool**.

Step 4 In the **Create VLAN Pool** dialog box, fill in the fields as required, except as specified below:

a) For the **Allocation Mode** radio buttons, choose **Static Allocation**.

b) In **Encap Blocks** section, click **+**.

Step 5 In the **Create Ranges** dialog box, enter a unique range of VLANs and click **OK**.

Step 6 In the **Create VLAN Pool** dialog box, click **Submit**.

Creating an External Routed Domain Using the GUI

You must create an external routed domain that ties together the location (leaf node/path) of the device and the static VLAN pool that you created for route peering.

Step 1 On the menu bar, choose **FABRIC** > **Access Policies**.

Step 2 In the Navigation pane, right-click **Switch Policies** and choose **Configure Interface, PC, and VPC**.

Step 3 In the **Configure Interface, PC, and VPC** dialog box, to configure switch ports connected to Application Policy Infrastructure Controllers (APICs), perform the following actions:

a) Click the large **+** icon next to the switch diagram to create a new profile and configure VLANs for the APIC.
b) From the **Switches** field drop-down list, check the check boxes for the switches to which the APICs are connected.
c) In the **Switch Profile Name** field, enter a name for the profile.
d) Click the **+** icon to configure the ports.
e) Verify that in the **Interface Type** area, the **Individual** radio button is selected.
f) In the **Interfaces** field, enter the ports to which APICs are connected.
g) In the **Interface Selector Name** field, enter the name of the port profile.
h) In the **Interface Policy Group** field, click the **Create One** radio button.
i) In the **Attached Device Type** drop-down list, choose **External Routed Devices**.
j) For the **Domain** radio buttons, click the **Create One** radio button.
k) In the **Domain Name** field, enter the domain name.
l) If you have previously created a VLAN pool, then for the **VLAN** radio buttons, click the **Choose One** radio button. Otherwise, click the **Create One** radio button.

   If you are choosing an existing VLAN pool, in the **VLAN Pool** drop-down list, choose the VLAN pool.

   If you are creating a VLAN pool, in the **VLAN Range** field, enter the VLAN range.

m) Click **Save**, and click **Save** again.
n) Click **Submit**.

Creating an External Routed Network Using the GUI

The external routed network specifies the routing configuration in the Cisco Application Centric Infrastructure (ACI) fabric for route peering.

Step 1 On the menu bar, choose **Tenants** > **All Tenants**.

Step 2 In the Work pane, double click the tenant's name.

Step 3 In the Navigation pane, choose ***tenant_name*** > **Networking** > **External Routed Networks**.

Step 4 In the Work pane, choose **Actions** > **Create Routed Outside**.

Step 5 In the **Create Routed Outside** dialog box, fill in the fields as required, except as specified below:

a) For dynamic routing, put a check in either the **BGP** or **OSPF** check box.

   For open shortest path first (OSPF), fill in the additional OSPF-specific fields.

b) In the **Private Network** drop-down list, choose the private network with which the device will exchange routes.

c) In the **External Routed Domain** drop-down list, choose the external routed domain that you created for route peering.
d) In the **Nodes and Interfaces Protocol Profiles** section, click +.

Step 6 In the **Create Node Profile** dialog box, fill in the fields as required, except as specified below:
a) In the **Nodes** section, click +.

Step 7 In the **Select Node** dialog box, fill in the fields as required, except as specified below:
a) In the **Node ID** drop-down list, choose the node ID where the device is connected.
- For physical devices, the ID should be the node where the physical device is connected to the fabric.
- For virtual devices, the ID should be the node where the server hosting the virtual machine is connected.

b) In the **Router ID** field, enter a router ID that the ACI fabric will use for the routing protocol process.
c) If you are planning to use static routing between the ACI fabric and device, in the **Static Routes** section click +. Otherwise, go to step Step 10, on page 63.

Step 8 In the **Create Static Route** dialog box, fill in the fields as required, except as specified below:
a) In the **Prefix** section, enter a prefix for the static route.
b) In the **Next Hop Addresses** section, click +.
c) Enter the next hop IP address for the static route.
d) Click **Update**.

Step 9 Click **OK**.

Step 10 In the **Select Node** dialog box, click **OK**.

Step 11 If you are using BGP as the dynamic routing protocol with the device, in the **BGP Peer Connectivity Profiles** section, click +. Otherwise, go to step Step 14, on page 63.

Step 12 In the **Create Peer Connectivity Profile** dialog box, fill in the fields as required, except as specified below:
a) In the **Peer Address** field, enter a peer address, which should be an IP address on the device with which the BGP session will be established.

Step 13 In the **Create Peer Connectivity Profile** dialog box, click **OK**.

Step 14 In the **Interface Profiles** section, click +.

Step 15 In the **Create Interface Profile** dialog box, fill in the fields as required.
a) If you are using OSPF as the dynamic routing protocol, enter the OSPF profile information.

Step 16 In the **Interface** section, choose the **SVI** tab.

Step 17 In the **Interface** section, click +.

Step 18 In the **Select SVI Interface** dialog box, fill in the fields as required, except as specified below:
a) For the **Path Type** radio buttons, choose the type that matches how the device is connected to the fabric.
b) In the **Path** drop-down list, choose the path where the device is connected to the fabric.
- For physical devices, this is the path where the physical device is connected to the fabric.
- For virtual devices, this is the path where the server that is hosting the virtual machine is connected.

c) In the **Encap** field, specify the encapsulation VLAN.
d) In the **IP Address** field, specify the IP address to use on the fabric SVI interface.
e) In the **MTU (bytes)** field, specify the maximum transmission unit size, in bytes.

The default value is "inherit", which uses a default value of "9000" on the ACI and typically a default value of "1500" on the remote device. Having different MTU values can cause issues when peering between the ACI and the remote device. If the remote device's MTU value is set to "1500", then set the MTU value on the remote device's `L3Out` object to "9000" to match the ACI's MTU value.

Step 19 Click **OK**.

Step 20 In the **Create Interface Profile** dialog box, click **OK**.

Step 21 In the **Create Node Profile** dialog box, click **OK**.

Step 22 In the **Create Routed Outside** dialog box, click **Next**.

Step 23 In the **External EPG Networks** section, click **+**.

Step 24 In the **Create External Network** dialog box, fill in the fields as required, except as specified below:

a) In the **Subnet** section, click **+**.

Step 25 In the **Create Subnet** dialog box, fill in the fields as required, except as specified below:

a) In the **IP Address** field, enter the IP address or subnet mask.

The subnet mask is equivalent to a network statement that is defined in a traditional routing protocol configuration.

Step 26 Click **OK**.

Step 27 (Optional) Create additional subnets as needed.

Step 28 In the **Create External Network** dialog box, click **OK**.

Step 29 In the **Create Routed Outside** dialog box, click **Finish**.

Creating a Router Configuration Using the GUI

As part of the routing protocol configuration, you must specify the router ID that will be used on the device.

Step 1 On the menu bar, choose **Tenants** > **All Tenants**.

Step 2 In the Work pane, double click the tenant's name.

Step 3 In the Navigation pane, choose ***tenant_name*** > **Services** > **L4-L7** > **Router configurations**.

Step 4 In the Work pane, in the **Router Configurations** table, click **+**.

Step 5 Enter an IP address to use as the router ID on the device.

Step 6 Click **Update**.

Creating a Service Graph Association Using the GUI

You must create a service graph association, which involves associating the external routed network policy and router configuration with a device selection policy.

Step 1 On the menu bar, choose **Tenants** > **All Tenants**.

Step 2 In the Work pane, double click the tenant's name.

Step 3 In the Navigation pane, choose **Tenant** ***tenant_name*** > **Services** > **L4-L7** > **Device Selection Policies** > ***device_selection_policy***.

Step 4 In the Navigation pane, choose ***tenant_name*** > **L4-L7 Services** > **Device Selection Policies** > ***device_selection_policy***. *device_selection_policy* is the device selection policy with which you want to perform route peering with the Cisco Application Centric Infrastructure (ACI) fabric.

Step 5 In the Work pane, in the properties section, in the **Router Config** drop-down list, choose the router configuration that you created for route peering.

Step 6 In the Navigation pane, expand the chosen device selection policy and choose the interface that will peer with the ACI fabric.

Step 7 In the Work pane, in the properties section, for the **Associated Network** radio buttons, choose **L3 External Network**.

Step 8 In the **L3 External Network** drop-down list, choose the external routed network that you created for route peering.

The following changes occur:

- The encapsulation VLAN for the interface that is associated with the external routed network is reprogrammed to match the VLAN that is configured as part of the external routed network interface profile
- The external routed network interface and routing protocol configuration is pushed to the leaf switch
- The routing protocol configuration is pushed to the device using the device package

Configuring Route Peering Using the NX-OS-Style CLI

This section provides example commands of using the NX-OS-style CLI to configure route peering.

Step 1 Enter the configure mode.

Example:

```
apic1# configure
```

Step 2 Enter the configure mode for a tenant.

Example:

```
apic1(config)# tenant 101
```

Step 3 Add a service graph and associate it with a contract.

Example:

```
apic1(config-tenant)# l4l7 graph g1 contract c1
```

Step 4 Add a node (service) that is associated with the device cluster.

Example:

```
apic1(config-graph)# service ASA_FW device-cluster-tenant 101 device-cluster ASA_FW1
```

Step 5 Under the service function, configure the consumer connector and provider cluster-interface.

Example:

```
apic1(config-service)# connector consumer cluster-interface provider
```

Step 6 Under the cluster-interface, specify the Layer 3 outside (l3extOut) and the endpoint group (l3extInstP) to be used for route peering with the service device, then exit the connector configuration mode.

Example:

```
apic1(config-connector)# l4l7-peer tenant 101 out l101 epg e101 redistribute bgp
apic1(config-connector)# exit
```

Step 7 Repeat step 5 and step 6 for the provider connector and consumer cluster-interface.

Example:

```
apic1(config-service)# connector provider cluster-interface consumer
apic1(config-connector)# l4l7-peer tenant 101 out l101 epg e101 redistribute bgp
apic1(config-connector)# exit
```

Step 8 (Optional) If you want to disassociate the endpoint group from the connector, use the **no l4l7-peer** command.

Example:

```
apic1(config-connector)# no l4l7-peer tenant 101 out l101 epg e101 redistribute bgp
```

Step 9 Create a router configuration policy under a tenant, supply a router ID for the peer Layer 4 to Layer 7 device, and exit back to the configuration mode.

Example:

```
apic1(config)# tenant 102
apic1(config-tenant)# rtr-cfg bgp1
apic1(config-router)# router-id 1.2.3.5
apic1(config-router)# exit
```

Step 10 Associate the router configuration policy with a particular service device and exit back to the tenant configuration mode.

Example:

```
apic1(config-tenant)# l4l7 graph g2 contract c2 subject http
apic1(config-graph)# service ASA_FW device-cluster-tenant 102 device-cluster ASA_FW2
apic1(config-service)# rtr-cfg bgp1
apic1(config-service)# exit
apic1(config-graph)# exit
```

Step 11 Associate a Layer 3 outside with a leaf interface and a VRF:

Example:

```
apic1(config-tenant)# external-l3 epg e101 l3out l101
apic1(config-tenant-l3ext-epg)# vrf member v101
apic1(config-tenant-l3ext-epg)# match ip 101.101.1.0/24
apic1(config-tenant-l3ext-epg)# exit
apic1(config-tenant)# exit
apic1(config)# leaf 101
apic1(config-leaf)# vrf context tenant 101 vrf v101 l3out l101
apic1(config-leaf-vrf)# ip route 101.101.1.0/24 99.1.1.2
apic1(config-leaf-vrf)# exit
apic1(config-leaf)# interface ethernet 1/10
apic1(config-leaf-if)# vrf member tenant 101 vrf v101 l3out l101
apic1(config-leaf-if)# vlan-domain member dom101
apic1(config-leaf-if)# no switchport
apic1(config-leaf-if)# ip address 99.1.1.1/24
apic1(config-leaf-if)# exit
apic1(config-leaf)# exit
```

For the complete configuration for Layer 3 external connectivity (Layer 3 outside) using the named mode, including routing protocols (BGP, OSPF) and route maps, see the *Cisco APIC NX-OS Style CLI Command Reference* document.

Note The external Layer 3 configuration in the CLI is available in two modes: basic mode and named mode. For a given tenant or VRF, user only one of these modes for all external Layer 3 configuration. Route peering is supported only in the named mode.

Troubleshooting Route Peering

If your Cisco Application Centric Infrastructure (ACI) fabric has a route peering or data traffic issue, there are several commands that you can run on ACI fabric leaf switches to troubleshoot the issue.

The following table provides troubleshooting commands that you can run in the switch shell on the fabric leaf switch.

Command	Description
`show ip route vrf all`	Displays all of the routes in a particular context, including dynamically learned routes.
`show ip ospf neighbor vrf all`	Displays Open Shortest Path First (OSPF) peering sessions with neighboring devices.
`show ip ospf vrf all`	Displays the run-time OSPF configuration in each context.
`show ip ospf traffic vrf all`	Examines OSPF traffic on each virtual routing and forwarding (VRF) context.
`show system internal policymgr stats`	Displays the contract filter rules on a particular leaf switch and examines the packet hit counts on the rules.

The following table provides a troubleshooting command that you can run in the **vsh_lc** shell.

Command	Description
`show system internal aclqos prefix`	Examines the IPv4 prefix association rules on a particular leaf switch and the traffic hit counts on the rules.

In addition to the shell commands, you can check the following things to help with troubleshooting:

- Health count on the device
- All of the faults and `NwIssues` under a particular tenant

Verifying the Leaf Switch Route Peering Functionality Using the CLI

You can use switch shell commands on the fabric leaf to verify the leaf switch configuration and route peering functionality.

Step 1 On the fabric leaf switch where the device is connected, verify that the SVI interface is configured:

```
fab2-leaf3# show ip interface vrf user1:global
IP Interface Status for VRF "user1:global"
vlan30, Interface status: protocol-up/link-up/admin-up, iod: 134,
  IP address: 1.1.1.1, IP subnet: 1.1.1.0/30
  IP broadcast address: 255.255.255.255
  IP primary address route-preference: 1, tag: 0
lo3, Interface status: protocol-up/link-up/admin-up, iod: 133,
  IP address: 10.10.10.1, IP subnet: 10.10.10.1/32
  IP broadcast address: 255.255.255.255
  IP primary address route-preference: 1, tag: 0

fab2-leaf3#
```

Interface vlan30 contains the SVI interface configuration and Interface lo3 contains the router ID specified in the external routed network configuration.

Step 2 Verify the Open Shortest Path First (OSPF) configuration on the fabric leaf switch:

```
fab2-leaf3# show ip ospf vrf user1:global

 Routing Process default with ID 10.10.10.1 VRF user1:global
 Stateful High Availability enabled
 Supports only single TOS(TOS0) routes
 Supports opaque LSA
 Table-map using route-map exp-ctx-2949120-deny-external-tag
 Redistributing External Routes from
   static route-map exp-ctx-st-2949120
   bgp route-map exp-ctx-proto-2949120
   eigrp route-map exp-ctx-proto-2949120
 Maximum number of non self-generated LSA allowed 100000
    (feature configured but inactive)
    Current number of non self-generated LSA 1
    Threshold for warning message 75%
    Ignore-time 5 minutes, reset-time 10 minutes
    Ignore-count allowed 5, current ignore-count 0
 Administrative distance 110
 Reference Bandwidth is 40000 Mbps
 SPF throttling delay time of 200.000 msecs,
   SPF throttling hold time of 1000.000 msecs,
   SPF throttling maximum wait time of 5000.000 msecs
 LSA throttling start time of 0.000 msecs,
   LSA throttling hold interval of 5000.000 msecs,
   LSA throttling maximum wait time of 5000.000 msecs
 Minimum LSA arrival 1000.000 msec
 LSA group pacing timer 10 secs
 Maximum paths to destination 8
 Number of external LSAs 0, checksum sum 0x0
 Number of opaque AS LSAs 0, checksum sum 0x0
 Number of areas is 1, 1 normal, 0 stub, 0 nssa
 Number of active areas is 1, 1 normal, 0 stub, 0 nssa
   Area (0.0.0.200)
        Area has existed for 00:17:55
        Interfaces in this area: 1 Active interfaces: 1
        Passive interfaces: 0  Loopback interfaces: 0
        SPF calculation has run 4 times
```

```
         Last SPF ran for 0.000273s
        Area ranges are
        Area-filter in 'exp-ctx-proto-2949120'
        Number of LSAs: 3, checksum sum 0x0
fab2-leaf3#
```

Step 3 Verify the OSPF neighbor relationship on the fabric leaf switch:

```
fab2-leaf3# show ip ospf neighbors vrf user1:global
 OSPF Process ID default VRF user1:global
 Total number of neighbors: 1
 Neighbor ID     Pri State            Up Time  Address         Interface
 10.10.10.2        1 FULL/BDR         00:03:02 1.1.1.2         Vlan30
fab2-leaf3#
```

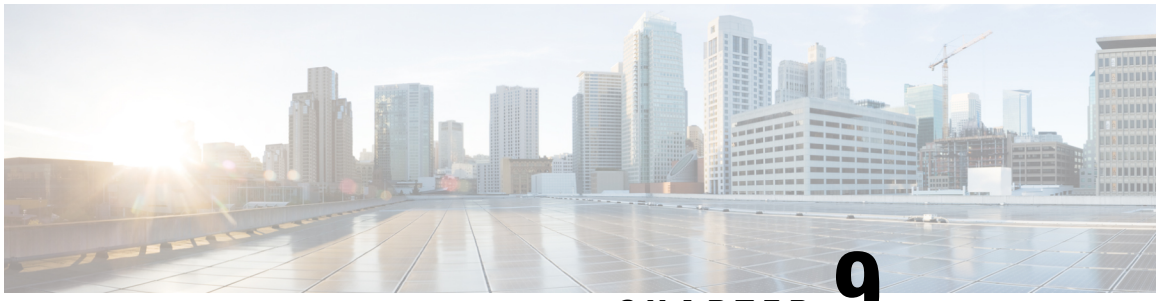
Step 4 Verify that the routes are being learned by the fabric leaf switch:

```
fab2-leaf3# show ip route vrf user1:global
IP Route Table for VRF "user1:global"
'*' denotes best ucast next-hop
'**' denotes best mcast next-hop
'[x/y]' denotes [preference/metric]
'%<string>' in via output denotes VRF <string>

1.1.1.0/30, ubest/mbest: 1/0, attached, direct
    *via 1.1.1.1, vlan30, [1/0], 00:26:50, direct
1.1.1.1/32, ubest/mbest: 1/0, attached
    *via 1.1.1.1, vlan30, [1/0], 00:26:50, local, local
2.2.2.0/24, ubest/mbest: 1/0
    *via 1.1.1.2, vlan30, [110/20], 00:06:19, ospf-default, type-2
10.10.10.1/32, ubest/mbest: 2/0, attached, direct
    *via 10.10.10.1, lo3, [1/0], 00:26:50, local, local
    *via 10.10.10.1, lo3, [1/0], 00:26:50, direct
10.122.254.0/24, ubest/mbest: 1/0
    *via 1.1.1.2, vlan30, [110/20], 00:06:19, ospf-default, type-2
fab2-leaf3#
```

Step 5 Verify that OSPF has been configured on the device, which is a Cisco ASAv in this example:

```
ciscoasa# show running-config
: Saved
:
: Serial Number: 9AGRM5NBEXG
: Hardware:   ASAv, 2048 MB RAM, CPU Xeon 5500 series 2133 MHz
:
ASA Version 9.3(1)
!
hostname ciscoasa
enable password 8Ry2YjIyt7RRXU24 encrypted
names
!
interface GigabitEthernet0/0
 nameif internalIf
 security-level 100
 ip address 2.2.2.1 255.255.255.0
!
interface GigabitEthernet0/1
 nameif externalIf
 security-level 50
 ip address 1.1.1.2 255.255.255.252
!
<<..>>
router ospf 1
 router-id 10.10.10.2
 network 1.1.1.0 255.255.255.252 area 200
```

```
 area 200
 log-adj-changes
 redistribute connected
 redistribute static
!
```



CHAPTER 9

Configuring Policy-Based Redirect

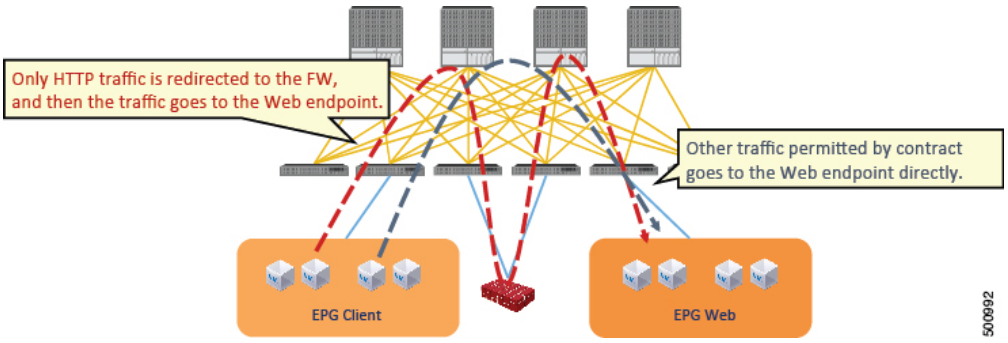
- About Policy-Based Redirect, on page 71
- About Multi-Node Policy-Based Redirect, on page 85
- About Symmetric Policy-Based Redirect, on page 85
- Policy Based Redirect and Hashing Algorithms, on page 86
- Policy-Based Redirect Resilient Hashing, on page 86
- PBR Support for Service Nodes in Consumer and Provider Bridge Domains , on page 88
- Policy-Based Redirect and Tracking Service Nodes, on page 88
- About Location-Aware Policy Based Redirect, on page 92
- Policy-Based Redirect and Service Graphs to Redirect All EPG-to-EPG Traffic Within the Same VRF Instance, on page 94

About Policy-Based Redirect

Cisco Application Centric Infrastructure (ACI) policy-based redirect (PBR) enables provisioning service appliances, such as firewalls or load balancers, as managed or unmanaged nodes without needing a Layer 4 to Layer 7 package. Typical use cases include provisioning service appliances that can be pooled, tailored to application profiles, scaled easily, and have reduced exposure to service outages. PBR simplifies the deployment of service appliances by enabling the provisioning consumer and provider endpoint groups to be all in the same virtual routing and forwarding (VRF) instance. PBR deployment consists of configuring a route redirect policy and a cluster redirect policy, and creating a service graph template that uses the route and cluster redirect policies. After the service graph template is deployed, use the service appliance by enabling endpoint groups to consume the service graph provider endpoint group. This can be further simplified and automated by using vzAny. While performance requirements may dictate provisioning dedicated service appliances, virtual service appliances can also be deployed easily using PBR.

The following figure illustrates the use case of redirecting specific traffic to the firewall:

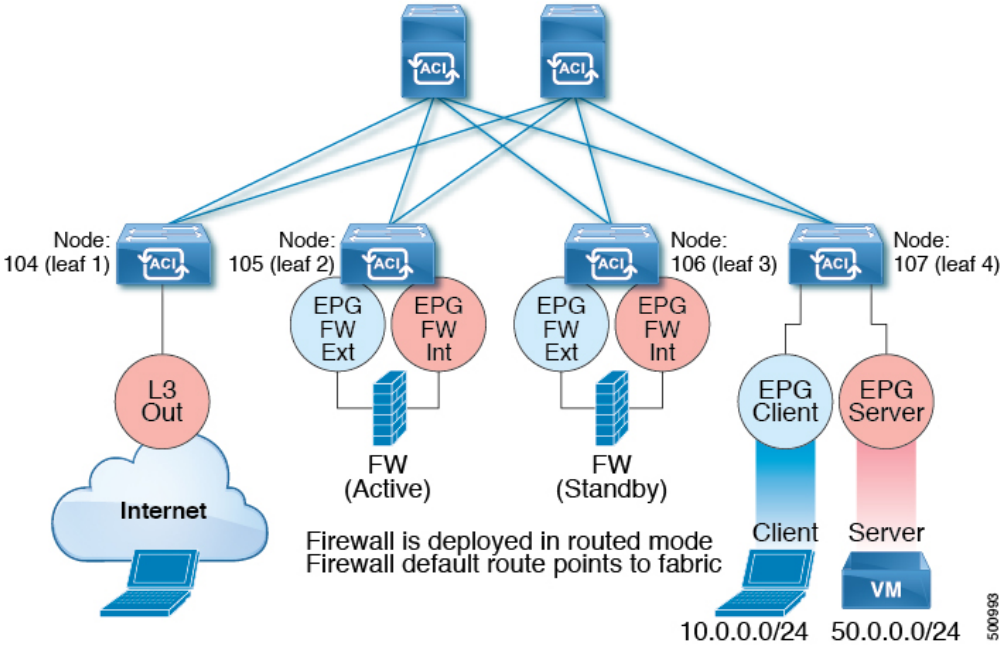
Figure 11: Use Case: Redirecting Specific Traffic to the Firewall



In this use case, you must create two subjects. The first subject permits HTTP traffic, which then gets redirected to the firewall. After the traffic passes through the firewall, it goes to the Web endpoint. The second subject permits all traffic, which captures traffic that is not redirected by the first subject. This traffic goes directly to the Web endpoint.

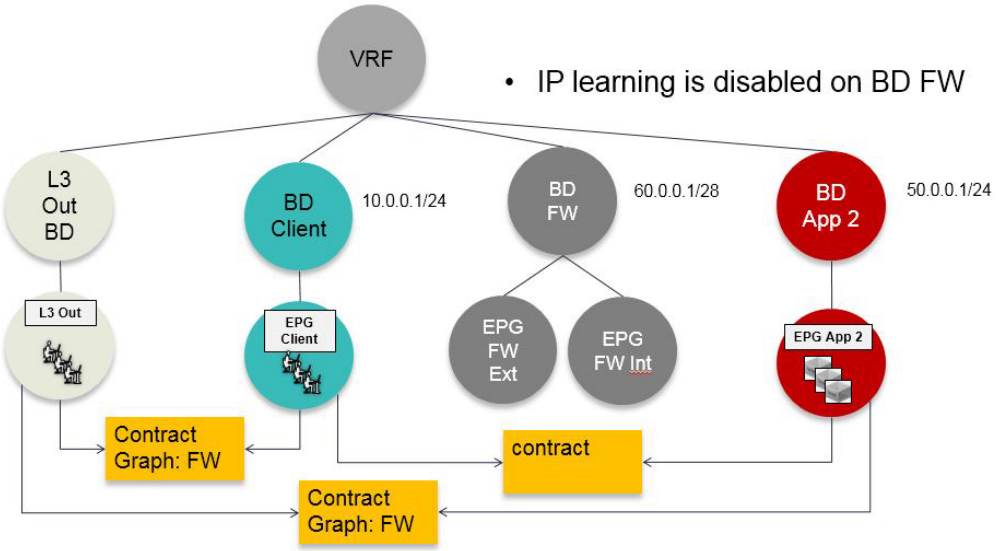
The following figure illustrates a sample ACI PBR physical topology:

Figure 12: Sample ACI PBR Physical Topology



The following figure illustrates a sample ACI PBR logical topology:

Figure 13: Sample ACI PBR Logical Topology



While these examples illustrate simple deployments, ACI PBR enables scaling up mixtures of both physical and virtual service appliances for multiple services, such as firewalls and server load balancers.

Guidelines and Limitations for Configuring Policy-Based Redirect

Observe the following guidelines and limitations when planning policy-based redirect (PBR) service nodes:

- A firewall (or a device that does not perform IP address translation) is inserted by using PBR for both directions.
- A load balancer (or a device that perorms IP address translation) is inserted by using unidirectional PBR. The destination IP address (VIP address or NAT'd IP address) for the other direction is owned by the device. The exception is Layer 2 direct server return, where the return traffic does not come back to the load balancer.
- The source MAC address of the packet can be rewritten because of the need to route the packet with PBR inside the fabric. The time-to-live (TTL) field in the IP address header will be decremented by as many times as the packet is routed within the fabric.
- Select the same action for both service legs. In other words, if you select the deny action for the internal service leg, you should also select the deny action for the external service leg.
- L3Out EPGs and regular EPGs can be consumer or provider EPGs.
- For a Cold Standby active/standby deployment, configure the service nodes with the MAC address of the active deployment. In a Cold Standby active/standby deployment, when the active node goes down, the standby node takes over the MAC address of active node.
- The next-hop service node IP address and virtual MAC address must be provided.
- Provision service appliances in a separate bridge domain. Starting with the Cisco Application Policy Infrastructure Controller (Cisco APIC) 3.1(1) release, it is not mandatory to provision service appliances in a separate bridge domain. To support this, Cisco Nexus 9300-EX and 9300-FX platform leaf switches are required.

- When downgrading from the Cisco APIC release 3.1 software, an internal code checks whether the policy-based redirect bridge domain uses the same bridge domain as a consumer or a provider. If it does, then the fault is disabled during the downgrade as such a configuration is not supported in earlier Cisco APIC versions.
- The service appliance, source, and bridge domain can be in the same VRF.
- For Cisco N9K-93128TX, N9K-9396PX, N9K-9396TX, N9K-9372PX, and N9K-9372TX switches, the service appliance must not be in the same leaf switch as either the source or destination endpoint group. For Cisco N9K-C93180YC-EX and N9K-93108TC-EX switches, the service appliance can be in the same leaf switch as either the source or destination endpoint group.
- PBR node interfaces are not supported on FEX host interfaces. A PBR node interface must be connected under leaf down link interface, not under FEX host interface. Consumer and Provider endpoints can be connected under FEX host interfaces.
- The service appliance can only be in a bridge domain.
- The contract offered by the service appliance provider endpoint group can be configured to `allow-all`, but traffic should be routed by the Cisco Application Centric Infrastructure (Cisco ACI) fabric.
- Starting with Cisco APIC release 3.1(1), if you use the Cisco Nexus 9300-EX and 9300-FX platform leaf switches, it is not necessary for you to have the endpoint dataplane learning disabled on policy-based redirect bridge domains. During service graph deployment, the endpoint dataplane learning will be automatically disabled only for policy-based redirect node EPG. If you use non-EX and non-FX platform leaf switches, you must have the endpoint dataplane learning disabled on policy-based redirect bridge domains. The policy-based redirect bridge domain must have the endpoint dataplane learning disabled.
- Multi-node policy-based redirect (multi-node PBR):
  - Supports up to three function nodes in a service graph that can be configured for policy-based redirect.
  - When using a multi-node PBR service chain, all the service devices have to be either in local leaf or they have to be connected to a remote leaf, but should not spread across both.
    - Supported topoloy:

      In this topology RL means remote leaf and LL means local leaf that is under main location, and not under remote leaf.

      - N1(LL)--N2(LL)--N3(LL) - All the devices are connected to local leafs not distributed across main location and remote leaf.
      - N1(RL)-N2(RL)--N3(RL) - All the devices are connected to remote leafs.

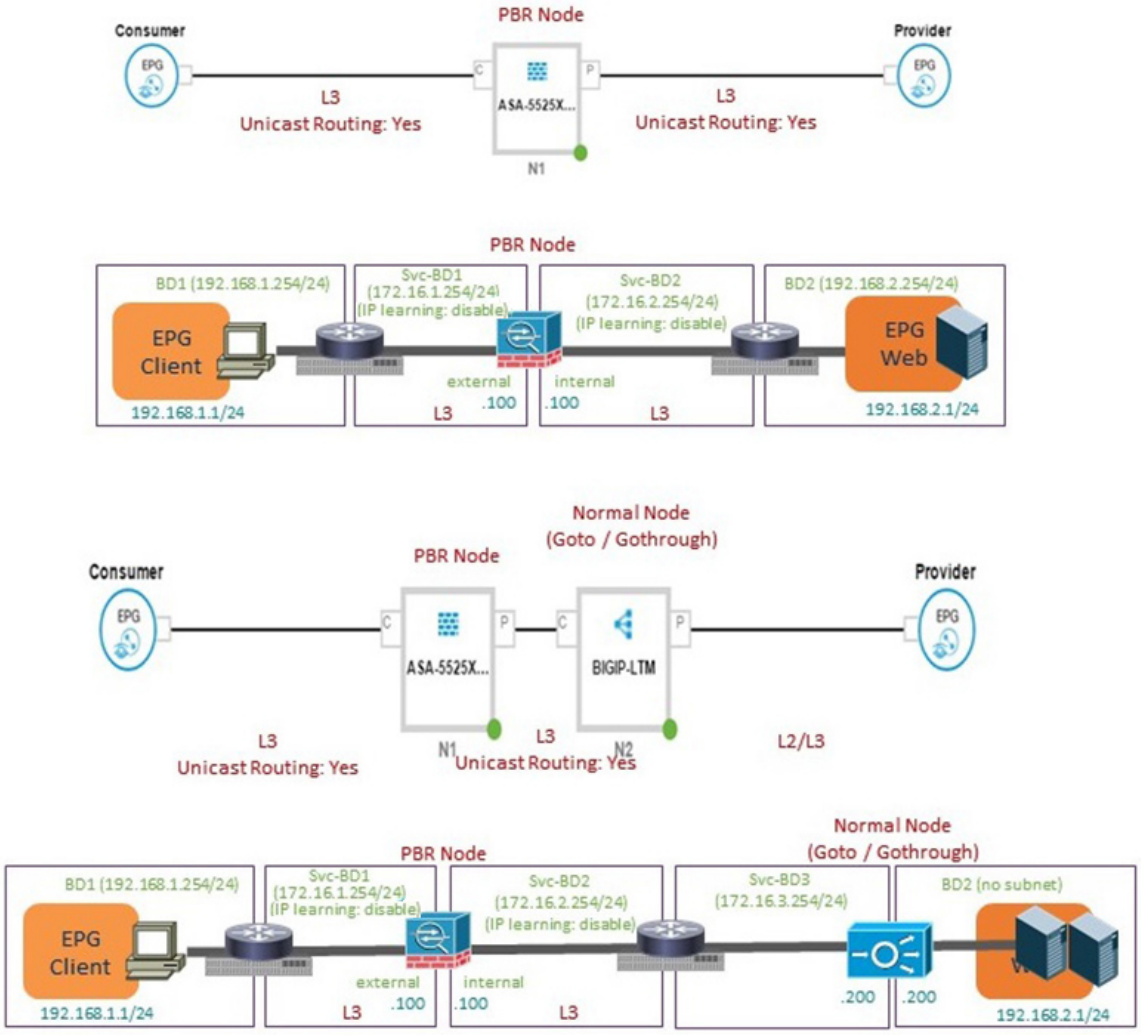
    - Topology not supported:
      - N1(LL)--N2(RL)--N3(LL) - Service devices are distributed across LL and RL.

  - Multi-node PBR Layer 3 destination guidelines for load balancers:
    - Layer 3 destination upgrade: The Layer 3 destination (VIP) parameter is enabled by default after the upgrade. No issues will occur from this because if the PBR policy was not configured on a specific service node (pre-3.2(1)), the node connector was treated as an Layer 3 destination and will continue to be in the new Cisco APIC version.

- Traffic does not always need to be destined to only consumer/provider
- In the forward direction, the traffic is destined to load balancer VIP
- In the reverse direction, if SNAT is enabled, the traffic is destined to the load balancer's internal leg
- In both directions, enable (check) Layer 3 destination (VIP) on the Logical Interface Context
- Enable (check) Layer 3 destination (VIP) in both directions to allow you to switch from SNAT to No-SNAT on the load balancer internal by configuring the PBR policy on the internal side
- If SNAT is disabled:
    - Reverse direction traffic is destined to consumer but not to load balancer internal leg (enable PBR policy on the internal leg)
    - Layer 3 destination (VIP) is not applicable in this case because a PBR policy is applied

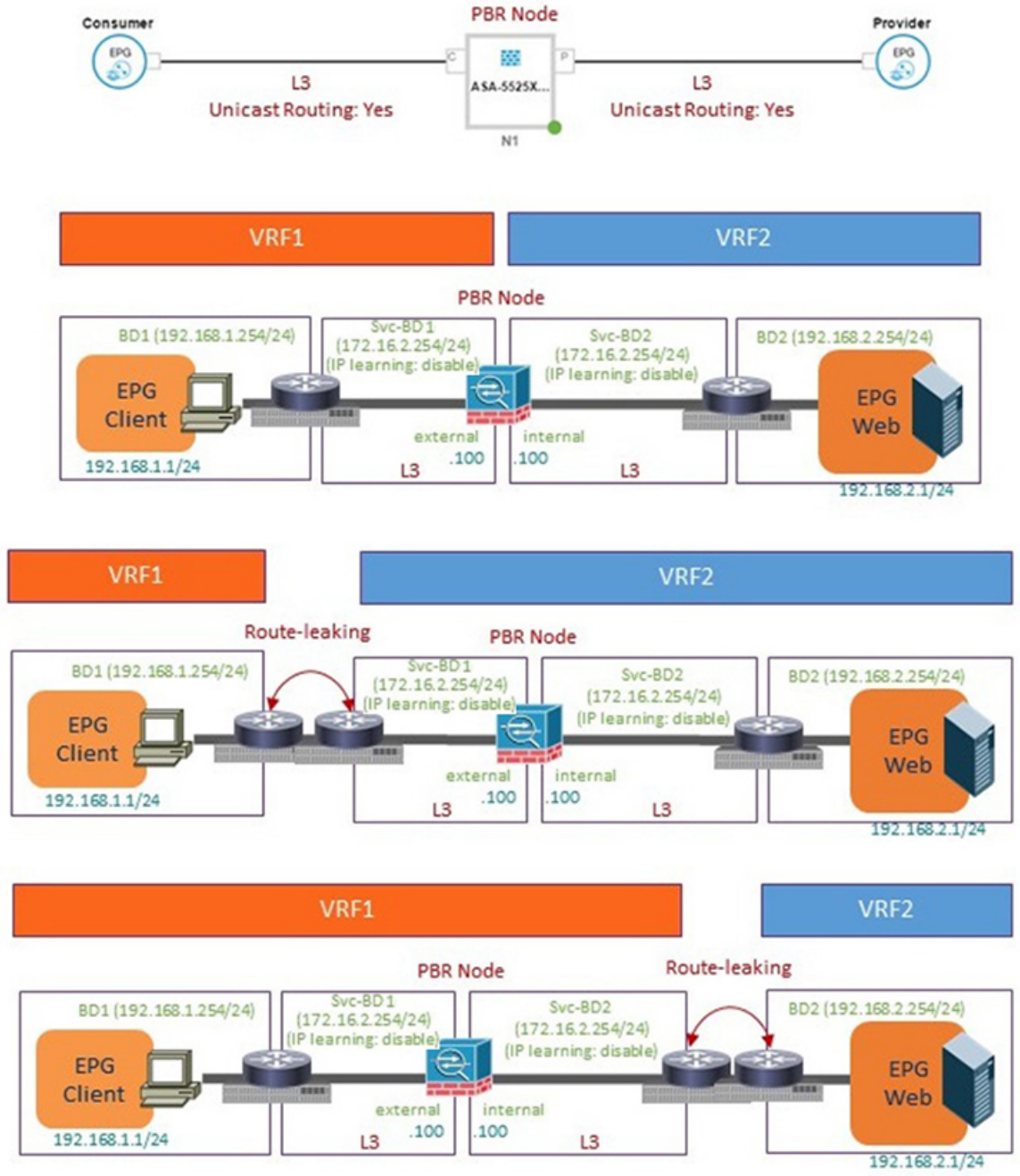
- Multicast and broadcast traffic redirection is not supported.
- If you change a redirect policy's destination to a different group, the Cisco APIC raises a fault due to the change and the policy's operational status becomes disabled. You must clear the fault to re-enable the policy.
- Supported policy-based redirect configurations in the same VRF instance include the following:

Figure 14: Supported Policy-based Redirect Configurations in the Same VRF Instance



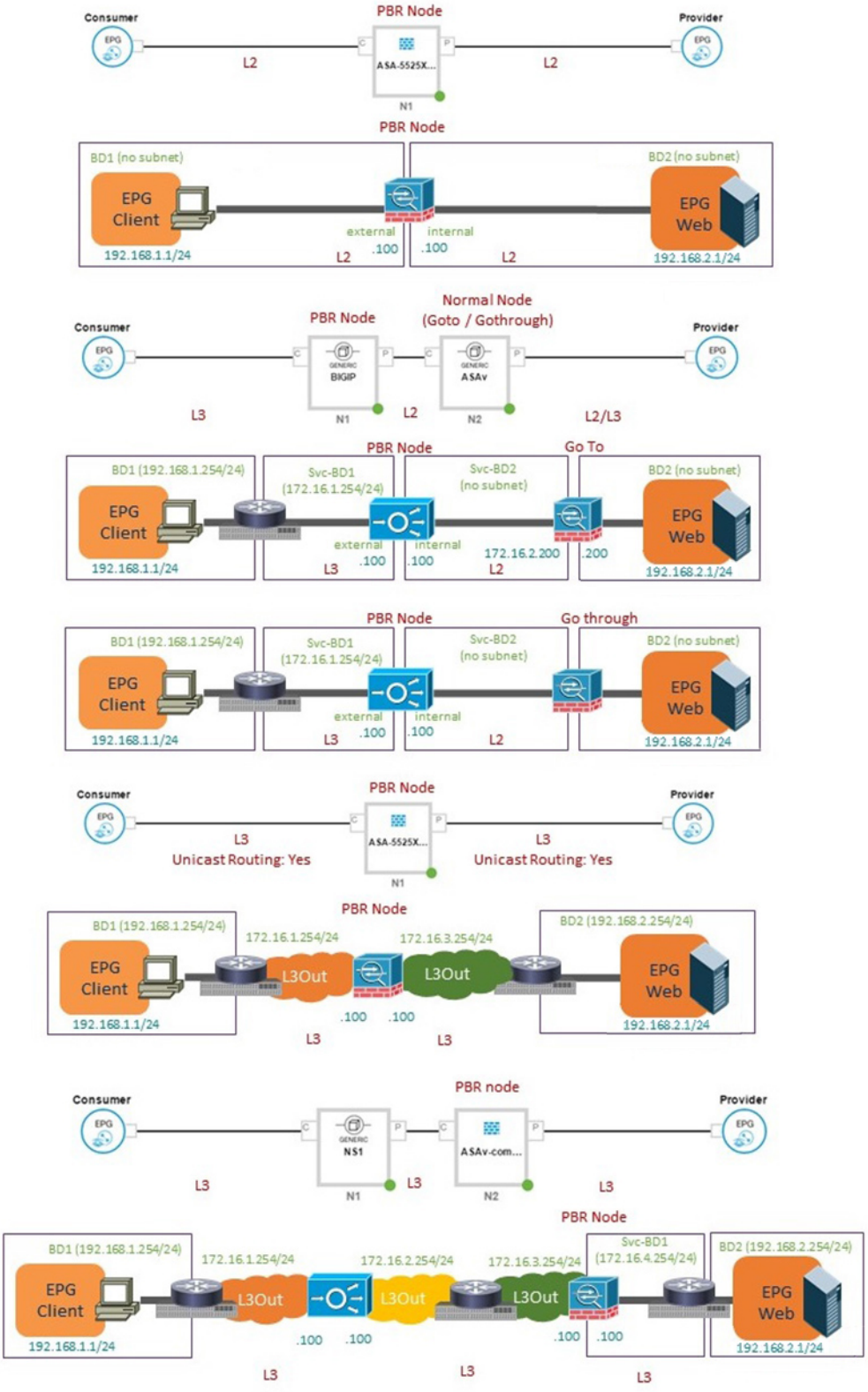
- Supported policy-based redirect configurations in a different VRF instance include the following:

Figure 15: Supported Policy-based Redirect Configurations in a Different VRF Instance



• Unsupported policy-based redirect configurations include the following:

Figure 16: Unsupported Policy-based Redirect Configurations



Configuring Policy-Based Redirect Using the GUI

The following procedure configures policy-based redirect (PBR) using the GUI.

Note The policy-based redirect feature is referred to as "policy-based routing" in the GUI.

Step 1 On the menu bar, choose **Tenants** > **All Tenants**.

Step 2 In the Work pane, double click the tenant's name.

Step 3 In the Navigation pane, choose **Tenant** ***tenant_name*** > **Services** > **L4-L7** > **Devices**.

Step 4 In the Work pane, choose **Actions** > **Create L4-L7 Devices**.

Step 5 In the **Create L4-L7 Devices** dialog box, complete the fields as required.

In the **General** section, the **Service Type** can be **Firewall** or **ADC**.

Note For L1/L2 PBR configuration, create the L4-L7 device in **Unmanaged** mode, and perform the following steps:

a. Select the **Service Type** as **Other**.

b. Select the **Device Type Physical** (cloud/virtual is not supported).

c. Select a physical domain.

d. Select the **Function Type L1** or **L2** as required.

e. Create external and internal concrete interfaces and port connectivity on the corresponding leafs.

f. Create Cluster interfaces by selecting the previously created concrete interfaces (leave VLAN encapsulation blank for dynamic assignments).

Note For static VLAN configuration, ensure external and internal legs have a different VLAN for L2, otherwise it is the same VLAN for L1.

Step 6 In the Navigation pane, choose **Tenant** ***tenant_name*** > **Services** > **L4-L7** > **Service Graph Templates**.

Step 7 In the Work pane, choose **Action** > **Create L4-L7 Service Graph Template**.

Step 8 In the **Create L4-L7 Service Graph Template** dialog box, perform the following actions:

a) In the **Graph Name** field, enter a name for the service graph template.

b) For the **Graph Type** radio buttons, click **Create A New Graph**.

c) Drag and drop the device that you created from the **Device Clusters** pane to between the consumer endpoint group and provider endpoint group. This creates the service node.

As of APIC Release 4.2(1), you can optionally repeat step c to include up to five (5) service node devices supporting PBR.

d) Select the following based on the service type of the device:

For Firewall, select **Routed** and continue with the steps below.

For ADC, select **One-Arm** or **Two-Arm** and continue with the steps below.

e) In the **Profile** drop-down list, select a function profile appropriate to the device. If no profiles exist, create one by following the instruction in the Creating a Function Profile Using the GUI.
f) Select the **Route Redirect** checkbox.
g) Click **Submit**.

The new service graph template appears in the Service Graph Templates table.

Step 9 In the Navigation pane, choose **Tenant** *tenant_name* > **Policies** > **Protocol** > **L4-L7 Policy Based Redirect**.

Step 10 In the Work pane, choose **Action** > **Create L4-L7 Policy Based Redirect**.

Step 11 In the **Create L4-L7 Policy Based Redirect** dialog box, complete the fields as required. This policy-based redirect policy is for the consumer connector.

Step 12 Create another policy-based redirect policy for the provider connector.

Step 13 In the Navigation pane, choose **Tenant *tenant_name*** > **Services** > **L4-L7** > **Service Graph Templates** > ***service_graph_template_name*** .

Choose the service graph template that you just created.

Step 14 Right click the service graph template and choose **Apply L4-L7 Service Graph Template**.

Step 15 In the **Apply L4-L7 Service Graph Template to EPGs** dialog box, perform the following actions:

a) In the **Consumer EPG/External Network** drop-down list, choose the consumer endpoint group.
b) In the **Provider EPG/External Network** drop-down list, choose the provider endpoint group.
c) For the **Contract** radio buttons, click **Create A New Contract**.
d) In the **Contract Name** field, enter a name for the contract.
e) Do not put a check in the **No Filter (Allow All Traffic)** check box.
f) On the **Filter Entries** table, click + to add an entry.
g) For the new filter entry, enter "IP" for the name, choose **IP** for the **Ether Type**, and click **Update**.
h) Click **Next**.
i) For the Consumer Connector **BD** drop-down list, choose the external bridge domain that connects to the consumer endpoint group. Select **No** for **IP Data-plane Learning**.
j) For the Consumer Connector **Redirect Policy** drop-down list, choose the redirect policy that you created for the consumer connector.
k) For the Consumer Connector **Cluster Interface** drop-down list, choose the consumer cluster interface.
l) For the Provider Connector **BD** drop-down list, choose the internal bridge domain that connects to the provider endpoint group. Select **No** for **IP Data-plane Learning**.
m) For the Provider Connector **Redirect Policy** drop-down list, choose the redirect policy that you created for the provider connector.
n) For the Provider Connector **Cluster Interface** drop-down list, choose the provider cluster interface.
o) Click **Next**.
p) Configure the parameters as necessary for the device.
q) Cick **Finish**.

Configuring Policy-Based Redirect Using the NX-OS-Style CLI

The example commands in this procedure include the route redirect, the cluster redirect, and the graph deployment. The device is created under tenant T1. The device is a Cisco ASA virtual device in managed mode; only unmanaged mode devices can be configured using the CLI.

Step 1 Create the device cluster.

Example:

```
l4l7 cluster name ifav-asa-vm-ha type virtual vlan-domain ACIVswitch service FW function go-to
    cluster-device Device2 vcenter ifav108-vcenter vm "ASAv_HA1"
    cluster-device Device1 vcenter ifav108-vcenter vm "ASAv_HA"
    cluster-interface provider
      member device Device1 device-interface GigabitEthernet0/1
        interface ethernet 1/45 leaf 102
        vnic "Network adapter 3"
        exit
      member device Device2 device-interface GigabitEthernet0/1
        interface ethernet 1/45 leaf 102
        vnic "Network adapter 3"
        exit
      exit
    cluster-interface failover_link
      member device Device1 device-interface GigabitEthernet0/8
        interface ethernet 1/45 leaf 102
        vnic "Network adapter 10"
        exit
      member device Device2 device-interface GigabitEthernet0/8
        interface ethernet 1/45 leaf 102
        vnic "Network adapter 10"
        exit
      exit
    cluster-interface consumer
      member device Device1 device-interface GigabitEthernet0/0
        interface ethernet 1/45 leaf 102
        vnic "Network adapter 2"
        exit
      member device Device2 device-interface GigabitEthernet0/0
        interface ethernet 1/45 leaf 102
        vnic "Network adapter 2"
        exit
      exit
    exit
exit
```

Step 2 Under tenant PBRv6_ASA_HA_Mode, deploy the PBR service graph instance.

Example:

```
tenant PBRv6_ASA_HA_Mode
  access-list Contract_PBRv6_ASA_HA_Mode_Filter
    match ip
    exit
```

Step 3 Create a contract for PBR with the filter match IP protocol. Under the subject, specify the Layer 4 to Layer 7 service graph name.

The contract offered by the service appliance provider endpoint group cannot be configured with the `allow-all` setting.

Example:

```
  contract Contract_PBRv6_ASA_HA_Mode
    scope tenant
    subject Subject
      access-group Contract_PBRv6_ASA_HA_Mode_Filter both
      l4l7 graph PBRv6_ASA_HA_Mode_Graph
      exit
    exit
  vrf context CTX1
```

```
  exit
vrf context CTX2
  exit
```

Step 4 Create a bridge domain for the client and server endpoint group. Both the client and server are in the same VRF instance.

Example:

```
bridge-domain BD1
  arp flooding
  l2-unknown-unicast flood
  vrf member CTX1
  exit
bridge-domain BD2
  arp flooding
  l2-unknown-unicast flood
  vrf member CTX1
  exit
```

Step 5 Create a separate bridge domain for the external and internal leg of the firewall.

PBR requires the learning of the source VTEP on remote leaf switches to be disabled, which is done using the **no ip learning** command.

Example:

```
bridge-domain External-BD3
  arp flooding
  no ip learning
  l2-unknown-unicast flood
  vrf member CTX1
  exit
bridge-domain Internal-BD4
  arp flooding
  no ip learning
  l2-unknown-unicast flood
  vrf member CTX1
  exit
```

Step 6 Create the application profile and specify the endpoint groups.

Example:

```
application AP1
  epg ClientEPG
    bridge-domain member BD1
    contract consumer Contract_PBRv6_ASA_HA_Mode
    exit
  epg ServerEPG
    bridge-domain member BD2
    contract provider Contract_PBRv6_ASA_HA_Mode
    exit
  exit
```

Step 7 Specify the default gateway for the bridge domains.

Example:

```
interface bridge-domain BD1
  ipv6 address 89:1:1:1::64/64
  exit
interface bridge-domain BD2
  ipv6 address 99:1:1:1::64/64
  exit

interface bridge-domain External-BD3
```

```
      ipv6 address 10:1:1:1::64/64
      exit
    interface bridge-domain Internal-BD4
      ipv6 address 20:1:1:1::64/64
      exit
```

Step 8 Import the device from tenant T1.

Example:

```
l4l7 cluster import-from T1 device-cluster ifav-asa-vm-ha
```

Step 9 Create the service graph using the service redirect policy.

Example:

```
l4l7 graph PBRv6_ASA_HA_Mode_Graph contract Contract_PBRv6_ASA_HA_Mode
      service N2 device-cluster-tenant T1 device-cluster ifav-asa-vm-ha mode FW_ROUTED svcredir
enable
      connector consumer cluster-interface consumer_PBRv6
          bridge-domain tenant PBRv6_ASA_HA_Mode name External-BD3
          svcredir-pol tenant PBRv6_ASA_HA_Mode name External_leg
          exit
      connector provider cluster-interface provider_PBRv6
          bridge-domain tenant PBRv6_ASA_HA_Mode name Internal-BD4
          svcredir-pol tenant PBRv6_ASA_HA_Mode name Internal_leg
          exit
        exit
    connection C1 terminal consumer service N2 connector consumer
    connection C2 terminal provider service N2 connector provider
    exit
```

Step 10 Create the service redirect policy for the external and internal legs. IPv6 addresses are used in this example; you can also specify IPv4 addresses using the same command.

Example:

```
svcredir-pol Internal_leg
    redir-dest 20:1:1:1::1 00:00:AB:CD:00:11
    exit
svcredir-pol External_leg
    redir-dest 10:1:1:1::1 00:00:AB:CD:00:09
    exit
exit
```

Verifying a Policy-Based Redirect Configuration Using the NX-OS-Style CLI

After you have configured policy-based redirect, you can verify the configuration using the NX-OS-style CLI.

Step 1 Show the running configuration of the tenant.

Example:

```
apic1# show running-config tenant PBRv6_ASA_HA_Mode svcredir-pol
# Command: show running-config tenant PBRv6_ASA_HA_Mode svcredir-pol
# Time: Wed May 25 00:57:22 2016
  tenant PBRv6_ASA_HA_Mode
    svcredir-pol Internal_leg
      redir-dest 20:1:1:1::1/32 00:00:AB:CD:00:11
      exit
```

```
    svcredir-pol External_leg
      redir-dest 10:1:1:1::1/32 00:00:AB:CD:00:09
      exit
    exit
```

Step 2 Show the running configuration of the tenant and its service graph.

Example:

```
apic1# show running-config tenant PBRv6_ASA_HA_Mode l4l7 graph PBRv6_ASA_HA_Mode_Graph
# Command: show running-config tenant PBRv6_ASA_HA_Mode l4l7 graph PBRv6_ASA_HA_Mode_Graph
# Time: Wed May 25 00:55:09 2016
  tenant PBRv6_ASA_HA_Mode
    l4l7 graph PBRv6_ASA_HA_Mode_Graph contract Contract_PBRv6_ASA_HA_Mode
     service N2 device-cluster-tenant T1 device-cluster ifav-asa-vm-ha mode FW_ROUTED svcredir enable

        connector consumer cluster-interface consumer_PBRv6

          bridge-domain tenant PBRv6_ASA_HA_Mode name External-BD3

          svcredir-pol tenant PBRv6_ASA_HA_Mode name External_leg

          exit

        connector provider cluster-interface provider_PBRv6

          bridge-domain tenant PBRv6_ASA_HA_Mode name Internal-BD4
          svcredir-pol tenant PBRv6_ASA_HA_Mode name Internal_leg
          exit
        exit
      connection C1 terminal consumer service N2 connector consumer
      connection C2 terminal provider service N2 connector provider
      exit
    exit
```

Step 3 Show the service graph configuration.

Example:

```
apic1# show l4l7-graph graph PBRv6_ASA_HA_Mode_Graph
Graph           : PBRv6_ASA_HA_Mode-PBRv6_ASA_HA_Mode_Graph
Graph Instances : 1

Consumer EPg     : PBRv6_ASA_HA_Mode-ClientEPG
Provider EPg     : PBRv6_ASA_HA_Mode-ServerEPG
Contract Name    : PBRv6_ASA_HA_Mode-Contract_PBRv6_ASA_HA_Mode
Config status    : applied
Service Redirect : enabled

Function Node Name : N2
 Connector   Encap       Bridge-Domain  Device Interface      Service Redirect Policy
 ----------  ----------  ----------     --------------------  -----------------------
 consumer    vlan-241    PBRv6_ASA_HA_  consumer_PBRv6        External_leg
                         Mode-
                         External-BD3
 provider    vlan-105    PBRv6_ASA_HA_  provider_PBRv6        Internal_leg
                         Mode-
                         Internal-BD4
```

About Multi-Node Policy-Based Redirect

Multi-node policy-based redirect enhances PBR by supporting up to five nodes in a single service graph. You can configure which service node connector terminates the traffic and based on this configuration, the source and destination class IDs for the service chain are determined. In the multi-node PBR feature, policy-based redirection can be enabled on the consumer, provider, or both of the service node connectors. It can also be configured for the forward or reverse directions. If the PBR policy is configured on a service node connector, then that connector does not terminate traffic.

About Symmetric Policy-Based Redirect

Symmetric policy-based redirect (PBR) configurations enable provisioning a pool of service appliances so that the consumer and provider endpoint groups traffic is policy-based. The traffic is redirected to one of the service nodes in the pool, depending on the source and destination IP equal-cost multi-path routing (ECMP) prefix hashing.

Note Symmetric PBR configurations require 9300-EX hardware.

Sample symmetric PBR REST posts are listed below:

```
Under fvTenant svcCont

<vnsSvcRedirectPol name="LoadBalancer_pool">
          <vnsRedirectDest name="lb1" ip="1.1.1.1" mac="00:00:11:22:33:44"/>
          <vnsRedirectDest name="lb2" ip="2.2.2.2" mac="00:de:ad:be:ef:01"/>
          <vnsRedirectDest name="lb3" ip="3.3.3.3" mac="00:de:ad:be:ef:02"/>
</vnsSvcRedirectPol>

<vnsLIfCtx name="external">
          <vnsRsSvcRedirectPol tnVnsSvcRedirectPolName="LoadBalancer_pool"/>
          <vnsRsLIfCtxToBD tDn="uni/tn-solar/bd-fwBD">
</vnsLIfCtx>

<vnsAbsNode name="FW" routingMode="redirect">
```

Sample symmetric PBR NX-OS-style CLI commands are listed below.

The following commands under the tenant scope create a service redirect policy:

```
apic1(config-tenant)# svcredir-pol fw-external
apic1(svcredir-pol)# redir-dest 2.2.2.2 00:11:22:33:44:56
```

The following commands enable PBR:

```
apic1(config-tenant)# l4l7 graph FWOnly contract default
apic1(config-graph)# service FW svcredir enable
```

The following commands set the redirect policy under the device selection policy connector:

```
apic1(config-service)# connector external
apic1(config-connector)# svcredir-pol tenant solar name fw-external
```

Policy Based Redirect and Hashing Algorithms

Note This feature is available in the APIC Release 2.2(3x) release and going forward with APIC Release 3.1(1). It is not supported in APIC Release 3.0(x).

In Cisco APIC, Release 2.2(3x), Policy Based Redirect feature (PBR) supports the following hashing algorithms:

- Source IP address
- Destination IP address
- Source IP address, Destination IP address, and Protocol number (default configuration).

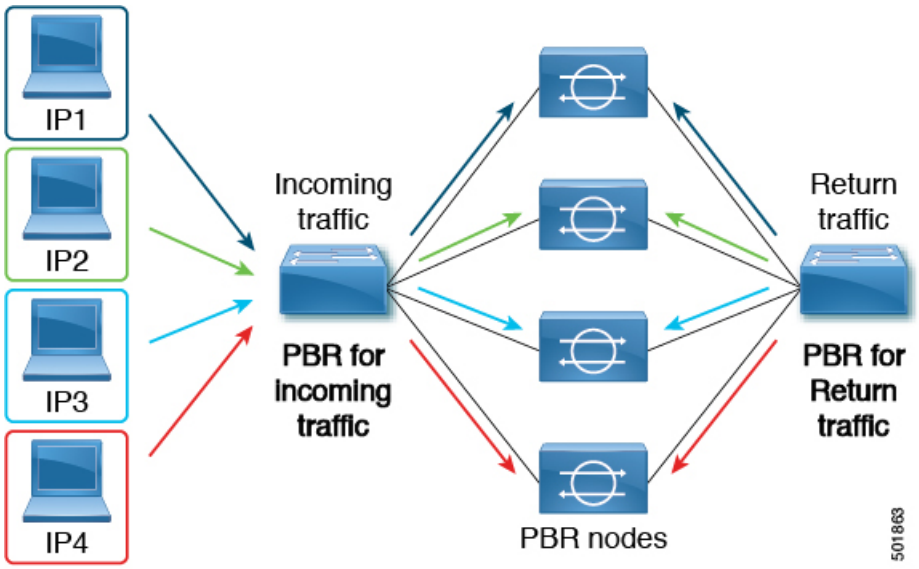
Policy-Based Redirect Resilient Hashing

In symmetric PBR, incoming and return user traffic uses the same PBR node in an ECMP group. If, however, one of the PBR nodes goes down/fails, the existing traffic flows are rehashed to another node. This can cause issues such as existing traffic on the functioning node being load balanced to other PBR nodes that do not have current connection information. If the traffic is traversing a stateful firewall, it can also lead to the connection being reset.

Resilient hashing is the process of mapping traffic flows to physical nodes and avoiding the rehashing of any traffic other than the flows from the failed node. The traffic from the failed node is remapped to a "backup" node. The existing traffic on the "backup" node is not moved.

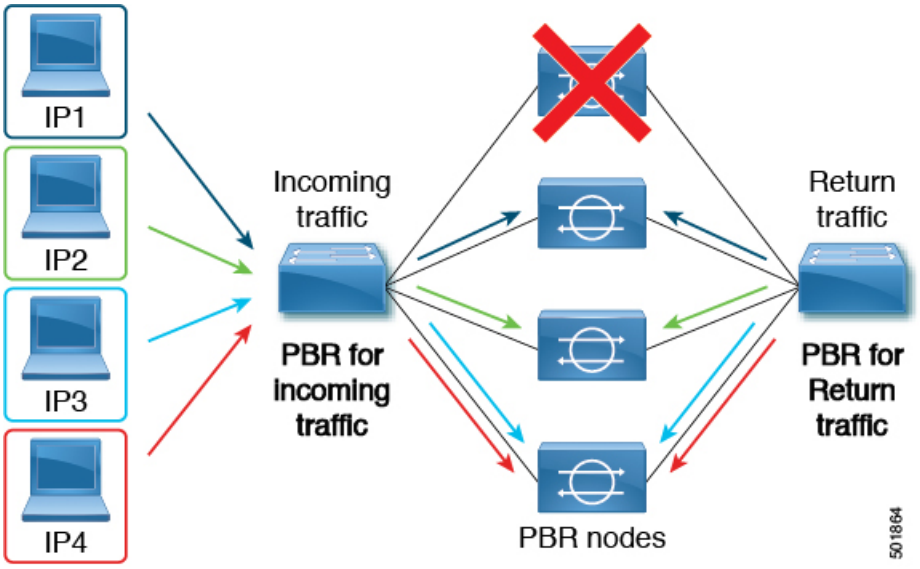
The image below shows the basic functionality of symmetric PBR with incoming and return user traffic using the same PBR nodes.

Figure 17: Symmetric PBR



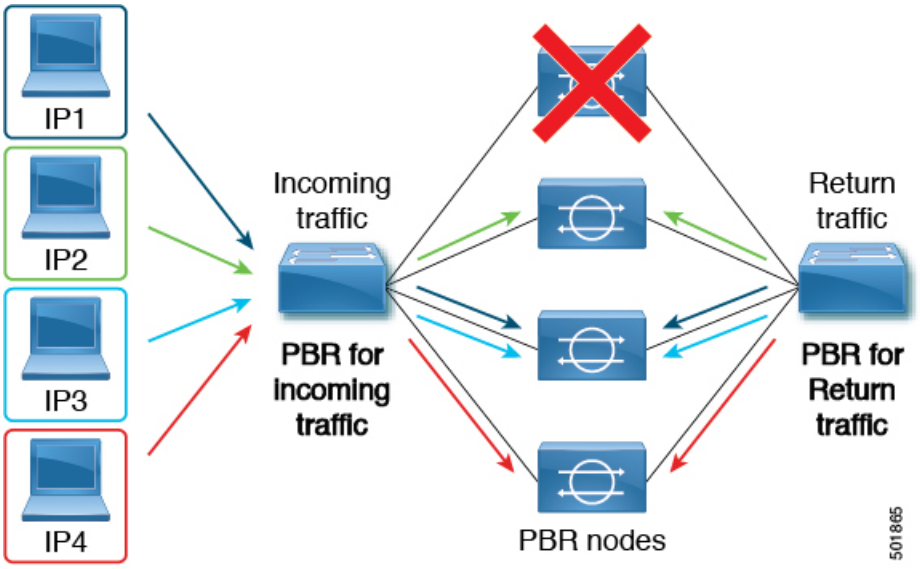
The next image shows what occurs when one of the PBR nodes is disabled or fails. The traffic for IP1 is rehashed to the next node and IP2 and IP3's traffic is load balanced to another PBR node. As stated earlier, this could lead to connectivity interruptions or delays if the other PBR nodes do not have the current connection information for IP2 and IP3 traffic.

Figure 18: Disabled/Failed PBR node without resilient hashing



The final image shows how this same use case is addressed when resilient hashing is enabled. Only the user traffic from the disabled/failed node is moved. All other user traffic remains on their respective PBR nodes.

Figure 19: Disabled/Failed PBR node with resilient hashing



If the node returns to service, the traffic flows rehashed from the failed node to the active node are returned to the reactivated node.

Note Adding or deleting PBR nodes from the ECMP group can cause all the traffic flows to be rehashed.

Enabling Resilient Hashing in L4-L7 Policy-Based Redirect

Before you begin

This task assumes that an L4-L7 Policy Based Redirect policy has been created.

Step 1 On the menu bar, choose **Tenants** > **All Tenants**.

Step 2 In the Work pane, double-click the tenant's name.

Step 3 In the Navigation pane, choose **Tenant** ***tenant_name*** > **Policies** > **Protocol** > **L4-L7 Policy Based Redirect** > ***L4-L7_PBR_policy_name*** .

Step 4 In the Work pane, check the **Resilient Hashing Enabled** check box.

Step 5 Click **Submit**.

PBR Support for Service Nodes in Consumer and Provider Bridge Domains

Starting with the Cisco APIC 3.1(1) release, bridge domains (BDs) that contain a consumer or provider also support service nodes. Therefore, you are not required to provision separate PBR bridge domains any longer.

The Cisco Nexus 9300-EX and 9300-FX platform leaf switches support this feature.

Policy-Based Redirect and Tracking Service Nodes

Beginning with the Cisco Application Policy Infrastructure Controller (APIC) 2.2(3) and 3.1(1) releases (but, excluding the 3.0 releases), the policy-based redirect feature (PBR) supports the ability to track service nodes. Tracking enables you to prevent redirection of traffic to a service node that is down. If a service node (PBR destination) is down, the PBR hashing can begin selecting an available PBR destination in a policy. This feature requires Cisco Nexus 9300-EX, -FX, or later platform leaf switches.

Service nodes can support dual IP address stacking. Therefore, this feature has the capability to track both IPv4 and IPv6 addresses at the same time. When both IPv4 and IPv6 addresses are "up," the PBR destination is marked as "up."

Switches internally use the Cisco IP SLA monitoring feature to support PBR tracking. The tracking feature marks a redirect destination node as "down" if the service node is not reachable. The tracking feature marks a redirect destination as node "up" if the service node resumes connectivity. When a service node is marked as "down," it will not be used to send or hash the traffic. Instead, the traffic will be sent or hashed to a different service node in the cluster of redirection destination nodes.

To avoid black holing of the traffic in one direction, you can associate a service node's ingress and egress redirect destination nodes with a redirection health policy. Doing so ensures that if either an ingress or egress redirection destination node is down, the other redirection destination node will also be marked as "down." Hence, both ingress and egress traffic gets hashed to a different service node in the cluster of the redirect destination nodes.

You can use the following protocols for tracking:

- ICMP (for Layer 3 PBR)
- TCP (for Layer 3 PBR)
- L2ping (for Layer 1/2 PBR)

Policy-Based Redirect and Threshold Settings for Tracking Service Nodes

The following threshold settings are available when configuring a policy-based redirect (PBR) policy for tracking service nodes:

- Threshold enabled or disabled: When the threshold is enabled, you an specify the minimum and maximum threshold percentages. Threshold enabled is required when you want to disable the redirect destination group completely and prevent any redirection. When there is no redirection, the traffic is directly sent between the consumer and the provider.
- Minimum threshold: The minimum threshold percentage specified. If the traffic goes below the minimum percentage, the packet is permitted instead of being redirected. The default value is 0.
- Maximum threshold: The maximum threshold percentage specified. Once the minimum threshold is reached, to get back to operational state, the maximum percentage must first be reached. The default value is 0.

Let us assume as an example that there are three redirect destinations in a policy. The minimum threshold is specified at 70% and the maximum threshold is specified at 80%. If one of the three redirect destination policies goes down, the percentage of availability goes down by one of three (or 33%), which is less than the minimum threshold. As a result, the minimum threshold percentage of the redirect destination group is brought down and traffic begins to get permitted instead of being redirected. Continuing with the same example, if the maximum threshold is 80%, to bring the redirect policy destination group back to the operational state, a percentage greater than the maximum threshold percentage must be reached.

Guidelines and Limitations for Policy-Based Redirect Tracking With Service Nodes

Follow these guidelines and limitations when using policy-based redirect (PBR) tracking with service nodes:

- A Cisco ACI Multi-Pod fabric setup is supported.
- A Cisco ACI Multi-Site setup is not supported.
- An L3Out is supported for the consumer and provider EPGs.
- TCP or ICMP protocol types are used to track the redirect destination nodes.
- PBR supports up to 100 trackable IP addresses in leaf switches and 200 trackable IP addresses in the Cisco Application Centric Infrastructure (ACI) fabric.

- PBR supports up to 1,000 service graph instances per Cisco ACI fabric.
- PBR supports up to 100 service graph instances per device.
- You can configure up to 40 service nodes per PBR policy.
- You can configure up to 3 service nodes per service chain.
- Shared services are supported with PBR tracking.
- The following threshold down actions are supported:
  - deny action
  - permit action
- If multiple PBR policies have the same PBR destination IP address in the same VRF instance, the policies must use the same IP SLA policy and health group for the PBR destination.

Configuring PBR and Tracking Service Nodes Using the GUI

Step 1 On the menu bar, click **Tenant** > ***tenant_name***. In the navigation pane, click **Policies** > **Protocol** > **L4-L7 Policy Based Redirect** .

Step 2 Right-click **L4 –L7 Policy Based Redirect**, and click **Create L4–L7 Policy Based Redirect**.

Step 3 In the **Create L4–L7 Policy Based Redirect** dialog box, perform the following actions:

a) In the **Name** field, enter a name for the PBR policy.

b) In the dialog box, choose the appropriate settings to configure the hashing algorithm, IP SLA Monitoring Policy, and other required values.

Note Destination groups that share destinations must have same IP SLA monitoring policy configured.

c) In the threshold setting fields, specify the settings as appropriate and if desired.

d) Expand **Destinations** to display **Create Destination of Redirected Traffic**.

e) In the **Create Destination of Redirected Traffic** dialog box, enter the appropriate details including the **IP** address and the **MAC address** fields.

The fields for IP address and Second IP address are provided where you can specify IPv4 and/or IPv6 addresses.

Note This field is not mandatory. Use it if the L4-L7 device has multiple IP addresses and you want ACI to verify both of them.

If both the **IP** and **Second IP** parameters are configured, both must be up in order to mark the PBR destination as "UP".

f) In the **Redirect Health Group** field, associate an existing health group or create a new health group, as appropriate. Click **OK**.

Note Destination groups that share destinations must have same health group configured.

g) In the **Create L4–L7 Policy Based Redirect** dialog box, click **Submit**.

The L4-L7 Policy Based Redirect and tracking of service nodes is configured after binding the redirect health group policy to the L4-L7 PBR policy and the settings to track the redirect destination group are enabled.

Configuring a Redirect Health Group Using the GUI

Step 1 On the menu bar, click **Tenant** > **tenant_name**. In the navigation pane, click **Policies** > **Protocol** > **L4-L7 Redirect Health Groups**.

Step 2 Right-click **L4 –L7 Redirect Health Groups**, and choose **Create L4–L7 Redirect Health Group**.

Step 3 In the **Create L4–L7 Redirect Health Group** dialog box, perform the following actions:

a) In the **Name** field, enter a name for the Redirect Health Group policy.

b) In the **Description** field, enter additional information if appropriate, and click **Submit**.

The Layer 4 to Layer 7 services redirect health policy is configured.

Configuring PBR to Support Tracking Service Nodes Using the REST API

Configure PBR to support tracking service nodes.

Example:

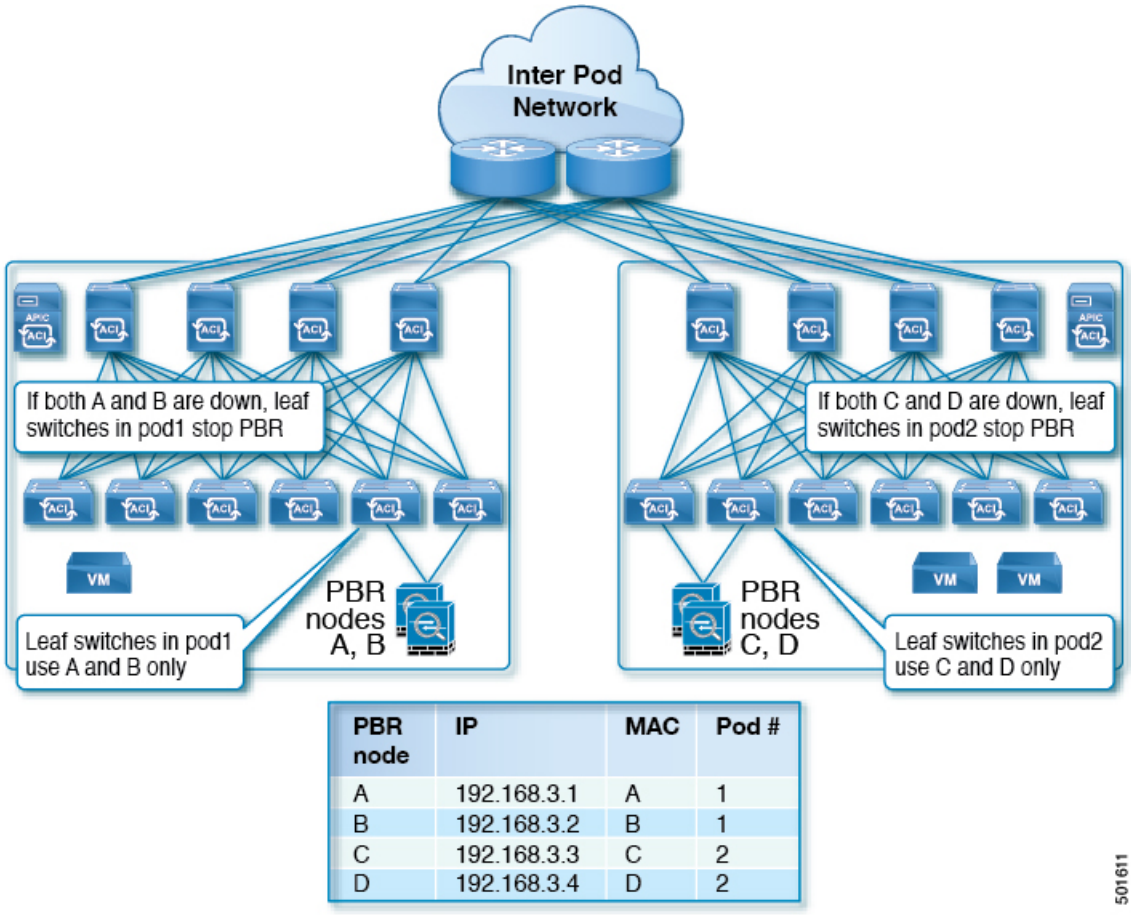
```
<polUni>
    <fvTenant name="t1" >
    <fvIPSLAMonitoringPol name="tcp_Freq60_Pol1" slaType="tcp" slaFrequency="60" slaPort="2222" />
    <vnsSvcCont>
        <vnsRedirectHealthGroup name="fwService1"/>
          <vnsSvcRedirectPol name="fwExt" hashingAlgorithm="sip" thresholdEnable="yes"
            minThresholdPercent="20" maxThresholdPercent="80">
            <vnsRedirectDest ip="40.40.40.100" mac="00:00:00:00:00:01">
               <vnsRsRedirectHealthGroup tDn="uni/tn-t1/svcCont/redirectHealthGroup-fwService1"/>
            </vnsRedirectDest>
            <vnsRsIPSLAMonitoringPol tDn="uni/tn-t1/ipslaMonitoringPol-tcp_Freq60_Pol1"/>
          </vnsSvcRedirectPol>
         <vnsSvcRedirectPol name="fwInt" hashingAlgorithm="sip" thresholdEnable="yes"
           minThresholdPercent="20" maxThresholdPercent="80">
             <vnsRedirectDest ip="30.30.30.100" mac="00:00:00:00:00:02">
                <vnsRsRedirectHealthGroup tDn="uni/tn-t1/svcCont/redirectHealthGroup-fwService1"/>
             </vnsRedirectDest>
             <vnsRsIPSLAMonitoringPol tDn="uni/tn-t1/ipslaMonitoringPol-tcp_Freq60_Pol1"/>
          </vnsSvcRedirectPol>
        </vnsSvcCont>
    </fvTenant>
</polUni>
```

About Location-Aware Policy Based Redirect

Location-Aware Policy Based Redirect (PBR) is now supported. This feature is useful in a multipod configuration scenario. Now there is pod-awareness support, and you can specify the preferred local PBR node. When you enable location-aware redirection, and Pod IDs are specified, all the redirect destinations in the Layer 4-Layer 7 PBR policy will have pod awareness. The redirect destination is programmed only in the leaf switches located in a specific pod.

The following image displays an example with two pods. PBR nodes A and B are in Pod 1 and PBR nodes C and D are in Pod 2. When you enable the location-aware PBR configuration, the leaf switches in Pod 1 prefer to use PBR nodes A and B, and the leaf switches in Pod 2 use PBR nodes in C and D. If PBR nodes A and B in Pod 1 are down, then the leaf switches in Pod 1 will start to use PBR nodes C and D. Similarly, if PBR nodes C and D in Pod 2 are down, the leaf switches in Pod 2 will start to use PBR nodes A and B.

Figure 20: An Example of Location Aware PBR Configuration with Two Pods



PBR node	IP	MAC	Pod #
A	192.168.3.1	A	1
B	192.168.3.2	B	1
C	192.168.3.3	C	2
D	192.168.3.4	D	2

Guidelines for Location-Aware PBR

Follow these guidelines when using location-aware PBR:

- The Cisco Nexus 9300 (except Cisco Nexus 9300–EX and 9300–FX) platform switches do not support the location-aware PBR feature.
- Use location-aware PBR for north-south firewall integration with GOLF host advertisement.

  Use location-aware PBR for a contract that is enforced on the same leaf nodes for incoming and returning traffic, such as an intra-VRF contract for external-EPG-to-EPG and an inter-VRF contract for EPG-to-EPG traffic. Otherwise, there can be a loss of traffic symmetry.
- If multiple PBR policies have the same PBR destination IP address in the same VRF, then all of the policies must either have Pod ID aware redirection enabled or Pod ID aware redirection disabled. The same (VRF, IP address) pair cannot be used in Pod ID aware redirection enabled and Pod ID aware redirection disabled policies at the same time. For example, the following configuration is not supported:
  - PBR-policy1 has PBR destination 192.168.1.1 in VRF A, Pod ID aware redirection enabled, and 192.168.1.1 is set to POD 1.
  - PBR-policy2 has PBR destination 192.168.1.1 in VRF A and Pod ID aware redirection disabled.

Configuring Location-Aware PBR Using the GUI

You must program two items for this feature to be enabled. Enable pod ID aware redirection and associate the Pod IDs with the preferred PBR nodes to program redirect destinations in the leaf switches located in the specific pods.

Step 1 On the menu bar, click **Tenant** > ***tenant_name*** . In the **Navigation** pane, click **Policies** > **Protocol** > **L4-L7 Policy Based Redirect** .

Step 2 Right-click **L4 –L7 Policy Based Redirect**, and click **Create L4–L7 Policy Based Redirect**.

Step 3 In the **Create L4–L7 Policy Based Redirect** dialog box, perform the following actions:

a) In the **Name** field, enter a name for the PBR policy.
b) In the **Enable Pod ID Aware Redirection** check the check box.
c) In the dialog box, choose the appropriate settings to configure the hashing algorithm, IP SLA Monitoring Policy, and other required values.
d) In the threshold setting fields, specify the settings as appropriate and if desired.
e) Expand **Destinations** to display **Create Destination of Redirected Traffic**.
f) In the **Create Destination of Redirected Traffic** dialog box, enter the appropriate details including the **IP** address and the **MAC address** fields.

   The fields for IP address and Second IP address are provided where you can specify an IPv4 address and an IPv6 address.

g) In the **Pod ID** field, enter the pod identification value.
h) In the **Redirect Health Group** field, associate an existing health group or create a new health group, as appropriate. Click **OK**.

   Create additional destinations of redirected traffic with different Pod IDs as required.

i) In the **Create L4–L7 Policy Based Redirect** dialog box, click **Submit**.

The L4-L7 location-aware PBR is configured.

Configuring Location-Aware PBR Using the REST API

You must configure two items to enable location-aware PBR and to program redirect destinations in the leaf switches located in the specific pods. The attributes that are configured to enable location-aware PBR in the following example are: programLocalPodOnly and podId.

Configure location-aware PBR.

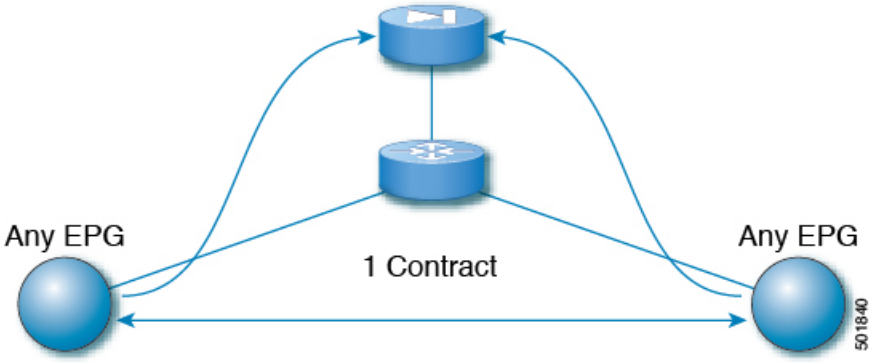
Example:

```
<polUni>
    <fvTenant name="coke" >
    <fvIPSLAMonitoringPol name="icmp_Freq60_Pol1" slaType="icmp" slaFrequency="60"/>
    <vnsSvcCont>
        <vnsRedirectHealthGroup name="fwService1"/>
          <vnsSvcRedirectPol name="fwExt" hashingAlgorithm="sip" thresholdEnable="yes"
minThresholdPercent="20" maxThresholdPercent="80" programLocalPodOnly="yes">
            <vnsRedirectDest ip="40.40.40.100" mac="00:00:00:00:00:01" podId="2">
               <vnsRsRedirectHealthGroup tDn="uni/tn-coke/svcCont/redirectHealthGroup-fwService1"/>
            </vnsRedirectDest>
            <vnsRsIPSLAMonitoringPol tDn="uni/tn-coke/ipslaMonitoringPol-icmp_Freq60_Pol1"/>
          </vnsSvcRedirectPol>
          <vnsSvcRedirectPol name="fwInt" hashingAlgorithm="dip" thresholdEnable="yes"
minThresholdPercent="20" maxThresholdPercent="80">
             <vnsRedirectDest ip="30.30.30.100" mac="00:00:00:00:00:02">
                <vnsRsRedirectHealthGroup tDn="uni/tn-coke/svcCont/redirectHealthGroup-fwService1"/>

             </vnsRedirectDest>
             <vnsRsIPSLAMonitoringPol tDn="uni/tn-coke/ipslaMonitoringPol-icmp_Freq60_Pol1"/>
          </vnsSvcRedirectPol>
        </vnsSvcCont>
    </fvTenant>
</polUni>
```

Policy-Based Redirect and Service Graphs to Redirect All EPG-to-EPG Traffic Within the Same VRF Instance

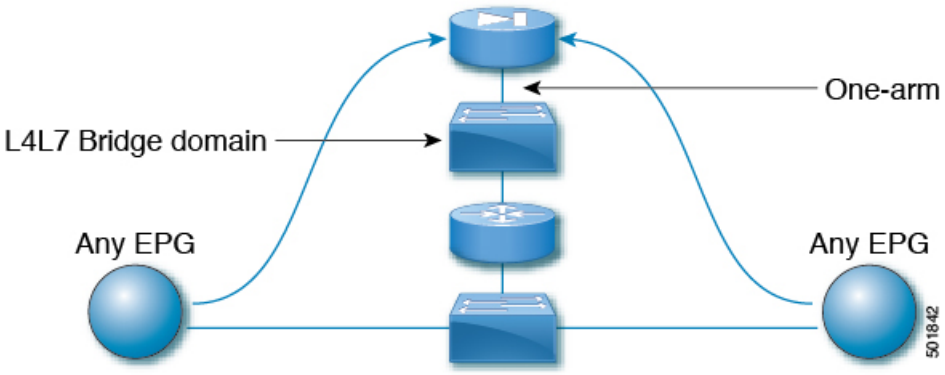
You can configure Cisco Application Centric Infrastructure (Cisco ACI) to forward all traffic from any endpoint group to any other endpoint group in the same VRF instance through a Layer 4 to Layer 7 device by configuring vzAny with service graph redirect. vzAny is a construct that represents all the endpoint groups under the same VRF instance. vzAny is sometimes referred to as "any EPG."

Figure 21: vzAny topology



Traffic between any endpoint group pair that is under the same VRF instance can be redirected to a Layer 4 to Layer 7 device, such as a firewall. You can also redirect traffic within the same bridge domain to a firewall. The firewall can filter traffic between any pair of endpoint groups, as illustrated in the following figure:

Figure 22: A firewall filtering traffic between any pair of EPGs

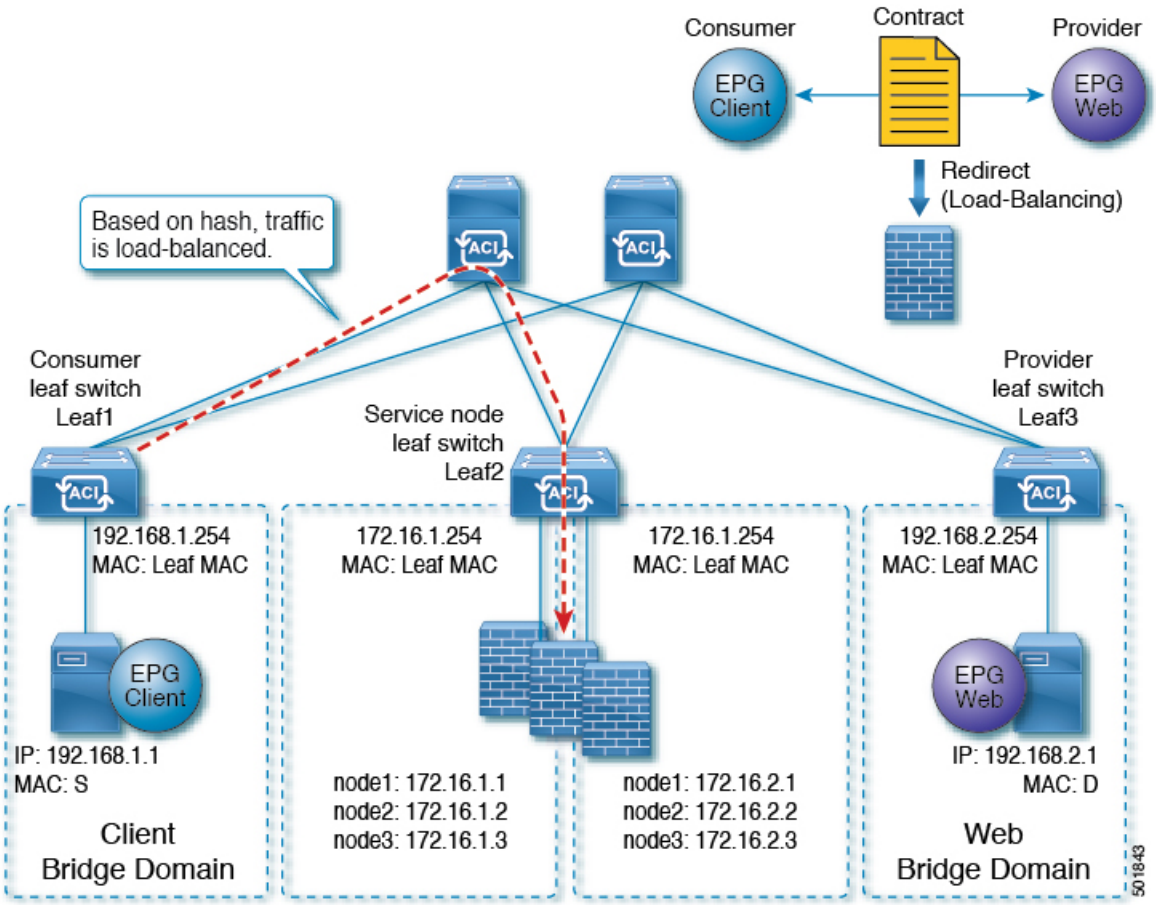


One use case of this functionality is to use Cisco ACI as a default gateway, but filter traffic through a firewall. With vzAny and a policy-based redirect policy, the security administrator manages the ACL rules and the network administrator manages routing and switching. Some of the benefits of this configuration include being able to use the Cisco Application Policy Infrastructure Controller's (Cisco APIC's) tools, such as endpoint tracking, first hop security with ARP inspection, or IP address source guard.

Applying a service graph with a policy-based redirect policy also enables the following functionality:

- Firewall clustering
- Firewall health tracking
- Location-aware redirection

Figure 23: Firewall clustering



Prior to the Cisco APIC 3.2 release, you could use vzAny as the consumer of a contract. Starting in the Cisco APIC 3.2 release, you can also use vzAny as the provider of a contract. This enhancement enables the following configurations:

- vzAny as the provider and vzAny as the consumer (policy-based redirect with one-arm only)
- vzAny as the provider and a regular endpoint group as the consumer (policy-based redirect and non-policy-based redirect case)

After you have applied a service graph with a policy-based redirect policy that redirects traffic using vzAny, if you want some traffic to bypass the firewall, such as for data backup traffic between two servers, you can create a more specific contract between the endpoint groups. For example, two endpoint groups can transmit traffic to one another directly over a given port. More specific rules win over the "any EPG to any EPG" redirect rule.

Guidelines and Limitations for Configuring a Policy-Based Redirect Policy with a Service Graph to Redirect All EPG-to-EPG Traffic Within the Same VRF Instance

The following guidelines and limitations apply when configuring a policy-based redirect policy with a service graph to redirect all EPG-to-EPG traffic within the same VRF instance:

- The Layer 4 to Layer 7 device and vzAny must belong to the same VRF instance.
- You must deploy the Layer 4 to Layer 7 device in one-arm mode.
- We generally recommend that you use a vzAny contract to enable PBR for many EPGs to many EPGs traffic instead of many EPGs consuming and providing the same contract. However, do not have a contract that has service graph attached as both the consumer and provider contract on the same EPG.

  This recommendation is due to a possible impact on changing a configuration on a contract that has many provider and consumer EPGs. If one configuration change on the Cisco Application Policy Infrastructure Controller (APIC) is related to multiple zoning-rule changes at the same time, the Cisco APIC needs time to finish programming the hardware of a given leaf node.
- vzAny configured with a multinode service graph might work, but this configuration has not been tested and is unsupported; use at your own risk.
- You can only deploy the Layer 4 to Layer 7 device in unmanaged mode.
- The use in conjunction with VRF leaking is not implemented. You cannot have vzAny of a VRF instance providing or consuming a vzAny contract of another VRF instance.
- You can have a contract between endpoint groups and vzAny in different tenants as long as they belong to the same VRF instance, such as if the VRF instance is in tenant **Common**.
- In a multipod environment, you can use vzAny as a provider and consumer.
- In a Cisco ACI Multi-Site environment, you cannot use vzAny as a provider and consumer across sites.

Configuring a Policy-Based Redirect Policy with a Service Graph to Redirect All EPG-to-EPG Traffic Within the Same VRF Instance

The following procedure configures a policy-based redirect policy with service graphs to redirect all EPG-to-EPG traffic within the same VRF instance:

Step 1 Create the service bridge domain that you will dedicate to the connectivity of the Layer 4 to Layer 7 device.

For information about creating a bridge domain, see the *Cisco APIC Basic Configuration Guide*.

On the **STEP 1 > Main** screen:

a) In the **VRF** drop-down list, choose the VRF instance that contains the endpoint groups.
b) In the **Forwarding** drop-down list, if you choose **Custom**, then in the **L2 Unknown Unicast** drop-down list, you can choose **Flood** if desired.

On the **STEP 2 > L3 Configurations** screen:

a) Ensure that there is a check in the **Unicast Routing** check box.
b) In the **Subnets** table, create a subnet.

The **Gateway IP** address must be in the same subnet as the IP address that you will give to the Layer 4 to Layer 7 device interface.

c) Remove the check from the **Endpoint Dataplane Learning** check box.

Step 2 Create the redirect policy.

a) In the **Navigation** pane, choose **Tenant tenant_name** > **Networking** > **Policies** > **Protocol** > **L4-L7 Policy Based Redirect**.
b) Right-click **L4-L7 Policy Based Redirect** and choose **Create L4-L7 Policy Based Redirect**.
c) In the **Name** field, enter a name for the policy.
d) In the **Destinations** table, click **+**.
e) In the **Create Destination of Redirected Traffic** dialog, enter the following information:

- **IP**—Enter the IP address that you will assign to the Layer 4 to Layer 7 device. The IP address must be in the same subnet as the IP address that you have given to the bridge domain.
- **MAC**—Enter the MAC address that you will assign to the Layer 4 to Layer 7 device. You should use a MAC address that is valid also upon failover of the Layer 4 to Layer 7 device. For example, in the case of a ASA firewall, this is called a "virtual MAC."

f) Enter any other desired values, then click **OK**.
g) In the **Create L4-L7 Policy Based Redirect** dialog, enter any other desired values, then click **Submit**.

Step 3 Create the Layer 4 to Layer 7 device with one concrete interface and one logical interface.

For information about creating a Layer 4 to Layer 7 device, see Configuring a Layer 4 to Layer 7 Services Device Using the GUI, on page 13.

Step 4 Create the service graph template with route redirect enabled.

a) In the **Navigation** pane, choose **Tenant tenant_name** > **Services** > **L4-L7** > **Service Graph Template**.
b) Right-click **Service Graph Template** and choose **Create Service Graph Template**.
c) In the **Name** field, enter a name for the service graph.
d) If you did not previously create the Layer 4 to Layer 7 device, in the **Device Clusters** pane, create the device.
e) Drag and drop the Layer 4 to Layer 7 device from the **Device Clusters** pane to in-between the consumer EPG and provider EPG.
f) For the **L4L7** radio buttons, click **Routed**.
g) Put a check in the **Routed Redirect** check box.
h) Click **Submit**.

Step 5 Apply the service graph to the vzAny (AnyEPG) endpoint group.

On the **STEP 1 > Contract** screen:

a) In the **Navigation** pane, choose **Tenant tenant_name** > **Services** > **L4-L7** > **Service Graph Template** > ***service_graph_name***.

service_graph_name is the service graph template that you just created.

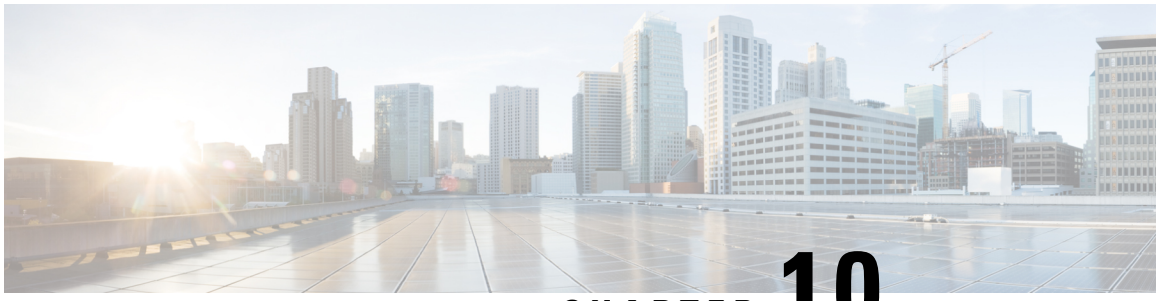
b) Right-click the service graph template and choose **Apply L4-L7 Service Graph Template**.
c) In the **Consumer EPG / External Network** drop-down list, choose the **AnyEPG** list item that corresponds to the tenant and VRF instance that you want to use for this use case.

For example, if the tenant is "tenant1" and the VRF instance is "vrf1," choose **tenant1/vrf1/AnyEPG**.

d) In the **Provider EPG / Internal Network** drop-down list, choose the same **AnyEPG** list item that you chose for the consumer EPG.
e) In the **Contract Name** field, enter a name for the contract.
f) Click **Next**.

On the **STEP 2 > Graph** screen:

a) For both **BD** drop-down lists, choose the Layer 4 to Layer 7 service bridge domain that you created in step 1.
b) For both **Redirect Policy** drop-down lists, choose the redirect policy that you created for this use case.
c) For the Consumer Connector **Cluster Interface** drop-down list, choose the cluster interface (logical interface) that you created in step 3.
d) For the Provider Connector **Cluster Interface** drop-down list, choose the same cluster interface (logical interface) that you created in step 3.
e) Click **Finish**.



CHAPTER 10

Configuring Direct Server Return

- About Direct Server Return, on page 101
- Example XML POST of Direct Server Return for Static Service Deployment, on page 105
- Direct Server Return for Static Service Deployment, on page 106
- Direct Server Return for Service Graph Insertion, on page 106
- Configuring the Citrix Server Load Balancer for Direct Server Return, on page 107
- Configuring a Linux Server for Direct Server Return, on page 107

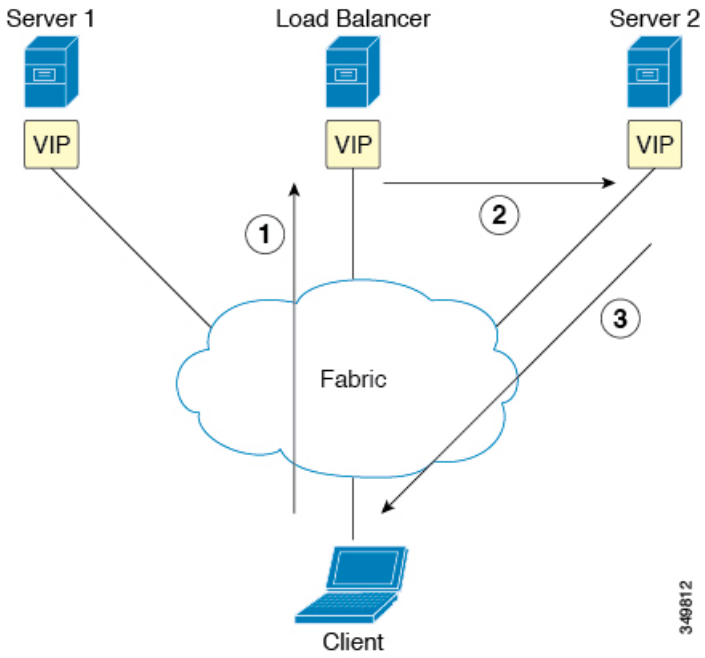
About Direct Server Return

The direct server return feature enables a server to respond directly to clients without having to go through the load balancer, which eliminates a bottleneck in the server-to-client path. In traditional load balancer deployments, the load balancer is in the path of the client-to-server communication: both the client-to-server request path and the server-to-client response path. While the amount of data in the requests from the client-to-server direction are relatively small, the server-to-client response traffic is much higher: approximately 10 times that of client-to-server request data. The load balancer in the path of this high volume response traffic becomes a bottleneck and adversely affects the communication.

For direct server return deployments, a virtual IP address is shared by the load balancer and server. Clients always address their request to the virtual IP address that is intended to reach the load balancer, and the direct response from the server-to-client use this virtual IP address as the source address. Cisco Application Centric Infrastructure (ACI) enabled with data-path learning of the IP source address poses problems when it learns the virtual IP address from the server-to-client traffic, leading to the disruption of Client-to-load balancer request traffic. To allow for the proper operation of a direct server return deployment, the ACI fabric must ensure that the request-response traffic between the communicating endpoints are delivered to their intended destination correctly. This requires that the data-path IP address learning on the leaf switches must be controlled in such a way that there is no interruption to client-to-load balancer, load balancer-to-server, and server-to-client traffic.

The following figure illustrates the data path in a direct server return deployment:

Figure 24: Direct Server Return High-Level Flow



1. The load balancer and all of the back-end servers are configured with the virtual IP address. The load balancer alone responds to Address Resolution Protocol (ARP) requests for this virtual IP address. After load balancing the client request, the load balancer re-writes the destination MAC address in the packet and forwards the MAC address to one of the back-end servers.
2. The virtual IP address is configured on the back-end server, but ARP is disabled to prevent back-end servers from responding to ARP requests for this virtual IP address.
3. The server sends the return traffic directly to the client, by-passing the load-balancer.

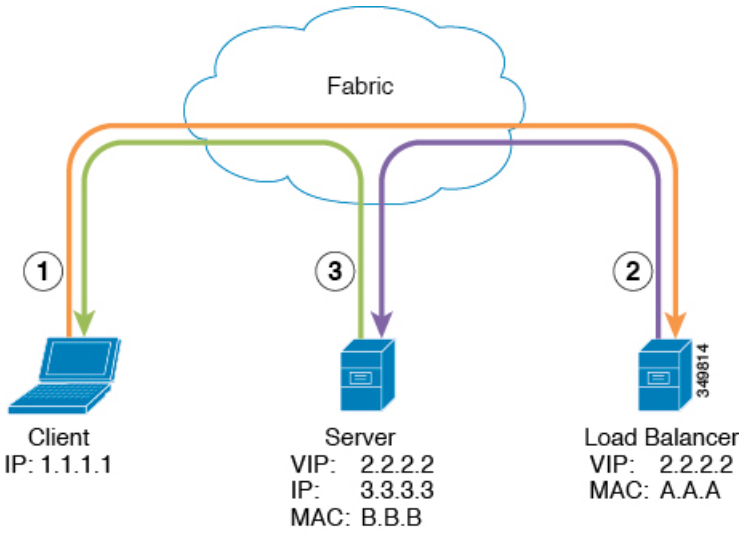
Layer 2 Direct Server Return

Layer 2 direct server return is the common or traditional deployment, also known as direct routing, SwitchBack, or nPath. In this deployment, the virtual IP address is shared by the load balancer and server. The load balancers and servers must be layer 2 adjacent. A layer 2 direct server return deployment has the following limitations:

- You lose flexibility in server placement
- You need an extra server configuration to suppress Address Resolution Protocol (ARP) responses to client virtual IP address requests
- Port selection is layer 3 and protocol dependent; port selection cannot happen at layer 2 (load balancer to server communication)

A layer 2 direct server return deployment has the following traffic flow:

Figure 25: Layer 2 Direct Server Return Traffic Flow



1. Client to load balancer

Source IP Address	1.1.1.1
Destination IP Address	2.2.2.2
Destination MAC Address	A.A.A

2. Load balancer to server

Source IP Address	1.1.1.1
Destination IP Address	2.2.2.2
Destination MAC Address	B.B.B

3. Server to client

Source IP Address	2.2.2.2
Destination IP Address	1.1.1.1
Destination MAC Address	MAC address of the default gateway

About Deploying Layer 2 Direct Server Return with Cisco Application Centric Infrastructure

The following information applies to deploying layer 2 direct server return with Cisco Application Centric Infrastructure (ACI):

- The virtual IP address (2.2.2.2) moves within the ACI fabric

    - The load balancer-to-server and server-to-client traffic with the same source virtual IP address (2.2.2.2)
    - The server-to-client traffic is routed; the traffic is addressed to the gateway MAC address in the fabric
    - The data-path learning of the source IP address from the server moves to the virtual IP address within the fabric

- There are no issues for the client IP address (1.1.1.1) appearing from difference sources
    - The client IP address appears as the source IP address from both the client and the load balancer in the fabric
    - The load balancer and server are layer 2 adjacent and the load balancer-to-server traffic is layer 2 forwarded
    - There is no data-path IP address learning from layer 2 forwarded traffic in the fabric
    - Even if the client IP address appears as the source IP address from the load balancer in the fabric, the client IP address is not learned

Guidelines and Limitations for Configuring Direct Server Return

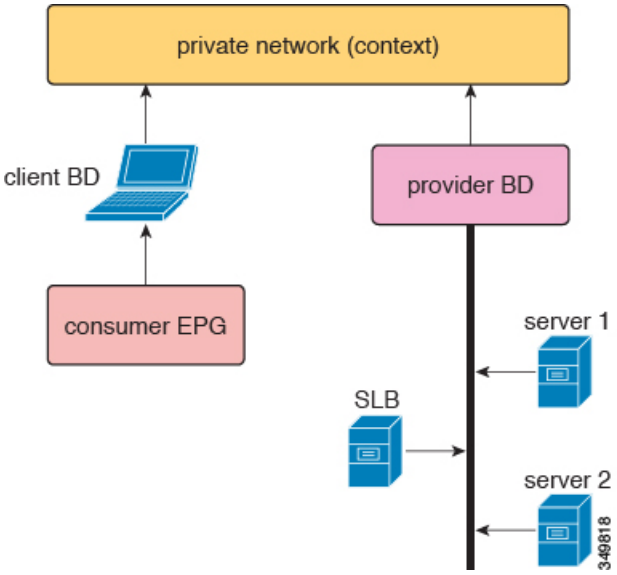
Follow these guidelines and limitations when deploying Direct Server Return:

- The VRF (where the VIP is deployed) must be set to "enforced" mode.
- The VRF must be set to "ingress" enforcement.
- Shared services are not supported for this configuration.
- EP Move Detection Mode: GARP-based Detection must be enabled for the bridge domain.
- Unicast routing must be enabled for the bridge domain.
- The EPG where the VIP is located must have a contract associated with it (the contract drives the configuration in hardware).
- VZAny contracts under VRF won't program L4-L7 VIP. The contract will still be allowed under EPG.
- Client to VIP traffic must always go through the proxy spine.
- The load balancer must be in one-armed mode.
- The server and load balancer EPG must be the same device or the load balancer EPG must be deployed on all server EPG ToRs.
- The server EPG and load balancer EPG must be in the same bridge domain.
- Configuring direct server return (DSR) with a Layer 4 to Layer 7 virtual IP (VIP) address under a microsegmented EPGs is not supported.

Supported Direct Server Return Configuration

The following figure illustrates the supported direct server return configuration:

Figure 26: Supported Direct Server Return Configuration



The following information applies to the supported configuration:

- The server load balancer and servers are in the same subnet and bridge domain
- The server load balancer should operate in 1 ARM mode; the inside and outside legs of server load balancer should point to the same bridge domain
- The consumer and provider endpoint groups should be under the same private network; no shared service configuration is supported

Example XML POST of Direct Server Return for Static Service Deployment

The following XML POST is an example of direct server return static service deployment:

```
<fvAp name="dev">
      <fvAEPg name="loadbalancer">
        <fvRsDomAtt tDn="uni/phys-{{tenantName}}"/>
        <fvRsBd tnFvBDName="lab"/>
        <fvVip addr="121.0.0.{{net}}" />
        <fvRsPathAtt tDn="topology/pod-1/paths-104/pathep-[eth1/1]" encap="vlan-33" />
        <fvRsProv tnVzBrCPName="loadBalancer"/>
        <fvRsCons tnVzBrCPName="webServer"/>
      </fvAEPg>
        <fvAEPg name="webServer">
          <fvRsDomAtt tDn="uni/phys-{{tenantName}}"/>
            <fvRsBd tnFvBDName="lab"/>
          <fvRsPathAtt tDn="topology/pod-1/paths-101/pathep-[eth1/1]" encap="vlan-34"/>
            <fvRsProv tnVzBrCPName="webServer"/>
        </fvAEPg>
      <fvAEPg name="client">
```

```
        <fvRsDomAtt tDn="uni/phys-{{tenantName}}"/>
        <fvRsBd tnFvBDName="lab"/>
        <fvRsPathAtt tDn="topology/pod-1/paths-103/pathep-[eth1/4]" encap="vlan-1114"/>
        <fvRsCons tnVzBrCPName="loadBalancer"/>
      </fvAEPg>
</fvAp>
```

The direct server return configuration is downloaded to all of the top-of-rack switches (ToRs) where the EPG with L4-L7 VIP is deployed or an EPG having contract with the EPG with L4-L7 VIP is deployed regardless of the contract direction. In the example, the direct server return virtual IP address configuration is downloaded to the ToR nodes-101, 103, and 104. Node-104 has a loadbalancer EPG with L4-L7 VIP configured and node-101 and 103 have webserver or client EPG, which has a contract to loadbalancer EPG.

All ToR having direct server return configuration downloaded does not learn L4-L7 VIP address from data-path and does not learn L4-L7 VIP address from other EPGs - even through ARP/GARP/ND. For example, a L4-L7 VIP address is only learned from loadbalancer EPG through a control plane. It helps to prevent to learn L4-L7 VIP mistakenly from a webserver EPG (for example, you forgot to suppress ARP on a webserver).

Direct Server Return for Static Service Deployment

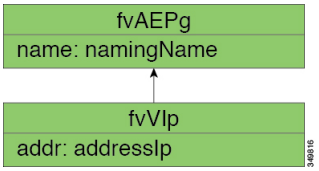
In the static service deployment mode, you configure the service flow by creating the appropriate application endpoint groups and contracts on a hop-by-hop basis.

Direct Server Return for Static Service Deployment Logical Model

You can configure the virtual IP addresses that are used by the load balancers by using the `fvVip` object under an application endpoint group (`fvAEPg`).

The following figure illustrates the logical model for static service deployment:

Figure 27: Static Service Deployment Logical Model



Direct Server Return for Service Graph Insertion

The Cisco Application Centric Infrastructure (ACI) provides automated service insertion by using vendor packages and service graphs. In this mode, the endpoint groups that are created for the service device legs, such as inside and outside endpoint groups, are created by the ACI without the operator configuration.

For service graph insertion, you must configure the virtual IP addresses under the appropriate logical interface context for the service device, as shown in the following example XML POST:

```
<vnsLDevCtx ctrctNameOrLbl="webCtrct"
  graphNameOrLbl="G1"
  nodeNameOrLbl="SLB">
    <vnsRsLDevCtxToLDev tDn="uni/tn-t1/lDevVip-InsiemeCluster"/>
    <vnsLIfCtx connNameOrLbl="inside">
        <vnsRsLIfCtxToBD tDn="uni/tn-t1/BD-t1BD1"/>
```

```
        <vnsRsLIfCtxToLIf tDn="uni/tn-t1/lDevVip-InsiemeCluster/lIf-inside"/>
    </vnsLIfCtx>
    <vnsLIfCtx connNameOrLbl="outside">
        <vnsRsLIfCtxToBD tDn="uni/tn-t1/BD-t1BD1"/>
        <vnsRsLIfCtxToLIf tDn="uni/tn-t1/lDevVip-InsiemeCluster/lIf-outside"/>
        <vnsSvcVip addr="9.9.9.9" />
        <vnsSvcVip addr="11.11.11.11" />
    </vnsLIfCtx>
</vnsLDevCtx>
```

The sample request configures two virtual IP addresses (9.9.9.9 and 11.11.11.11) on the outside leg of the server load balancer. The virtual IP address definition is under `LIfCtx` instead of being under an endpoint group as it is with a static direct server return configuration. This is because in the service graph case, operators do not have direct access to an endpoint group for the device legs, unlike with a static service deployment.

Direct Server Return Shared Layer 4 to Layer 7 Service Configuration

When the service device is configured in the common tenant or management tenant, the implicit model differs slightly. Instead of `vnsEPpInfo`, the service virtual IP address update managed object is created as a child of `vnsREPpInfo`. One `vnsSvcEpgCont` managed object is created per `vnsRsEPpInfo` to keep track of shared `SvcVip`s across tenants.

Configuring the Citrix Server Load Balancer for Direct Server Return

The following procedure provides an overview of how to configure the Citrix server load balancer for direct server return.

Step 1 Configure the virtual IP address on the backend server's loopback so that the backend server accepts the packets.

Step 2 Disable Address Resolution Protocol (ARP) reply for the virtual IP address on backend server.

Step 3 If necessary, disable the proxy port on services that are bound to the load balancing virtual server. The proxy port is disabled by default.

Step 4 Set the m parameter to "MAC" on the load balancing virtual server.

Step 5 Enable the USIP mode either globally or for each service.

Step 6 Enable the "L3", "USNIP", and "MBF" modes.

Step 7 Configure a route on the backend servers so that they can reach the Internet directly.

Configuring a Linux Server for Direct Server Return

The following procedure provides an overview of how to configure a Linux server for direct server return.

Step 1 Configure the virtual IP addresses on the loopback interfaces by creating the `/etc/sysconfig/network-scripts/ifcfg-lo` file in Centos with the following contents:

```
DEVICE=lo:1
IPADDRESS=10.10.10.99
NETMASK=255.255.255.255
NETWORK=10.10.10.99
BROADCAST=10.10.10.99
ONBOOT=yes
NAME=loopback
```

In this example, 10.10.10.99 is the virtual IP address.

Step 2 Set the arp_ignore and arp_announce settings in the server interface that is used to reply to client request:

```
echo 1 > /proc/sys/net/ipv4/conf/eth1/arp_ignore
echo 2 > /proc/sys/net/ipv4/conf/eth1/arp_announce
```

In this example, `eth1` is the server interface used to respond to client requests.

For more information about the ARP settings, see the following Linux Virtual Server wiki page:

http://kb.linuxvirtualserver.org/wiki/Using_arp_announce/arp_ignore_to_disable_ARP

CHAPTER 11

Configuring the Device and Chassis Manager

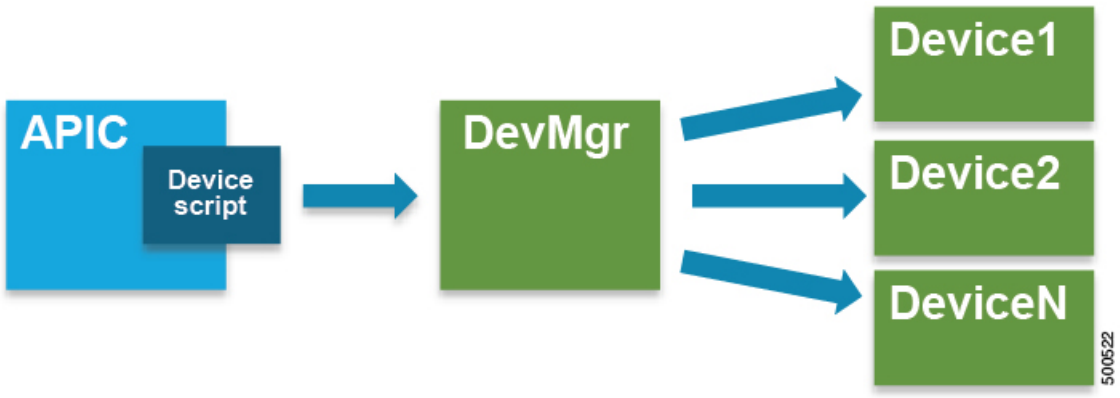
- About Device Managers and Chassis Managers, on page 109
- Device Manager and Chassis Manager Behavior, on page 112
- Creating Devices Using the GUI, on page 113
- Creating a Chassis Using the GUI, on page 113
- Example XML for Device Managers and Chassis Managers, on page 114
- Device and Chassis Callouts, on page 116

About Device Managers and Chassis Managers

A device manager serves as a single point of configuration for a set of clusters in a Cisco Application Centric Infrastructure (ACI) fabric. The administration or operational state is presented in the native GUI of the devices. A device manager handles configuration on individual devices and enables you to simplify the configuration in the Application Policy Infrastructure Controller (APIC). You create a template in device manager, then populate the device manager with instance-specific values from the APIC, which requires only a few values.

The following figure illustrates a device manager controlling multiple devices in a cluster:

Figure 28: Controlling Devices with a Device Manager

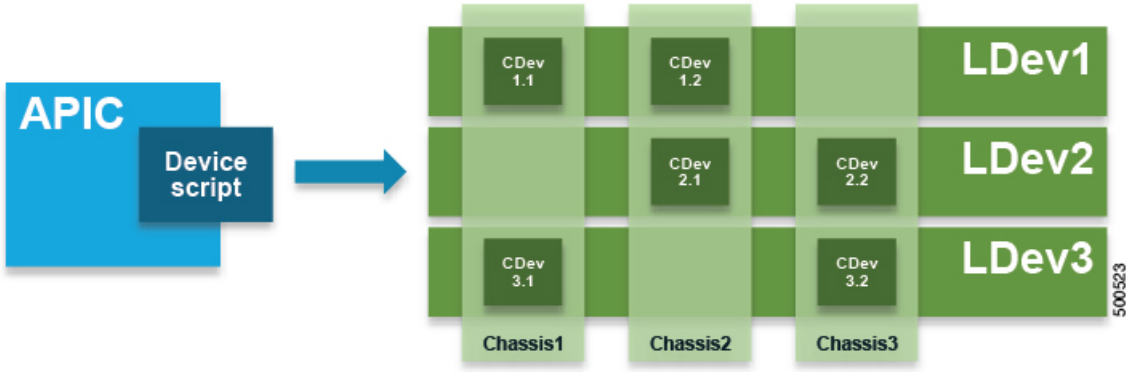


A chassis manager is a physical or virtual "container" of processing resources. A chassis manager supports a number of virtual service devices that are represented as `CDev` objects. A chassis manager handles networking, while `CDev` handles processing. A chassis manager enables the on-demand creation of virtual processing nodes.

For a virtual device, some parts of a service (specifically the VLANs) must be applied to the chassis rather than to the virtual machine. To accomplish this, the chassis management IP address and credentials must be included in callouts.

The following figure illustrates a chassis manager acting as a container of processing resources:

Figure 29: Controlling Devices with a Device Manager

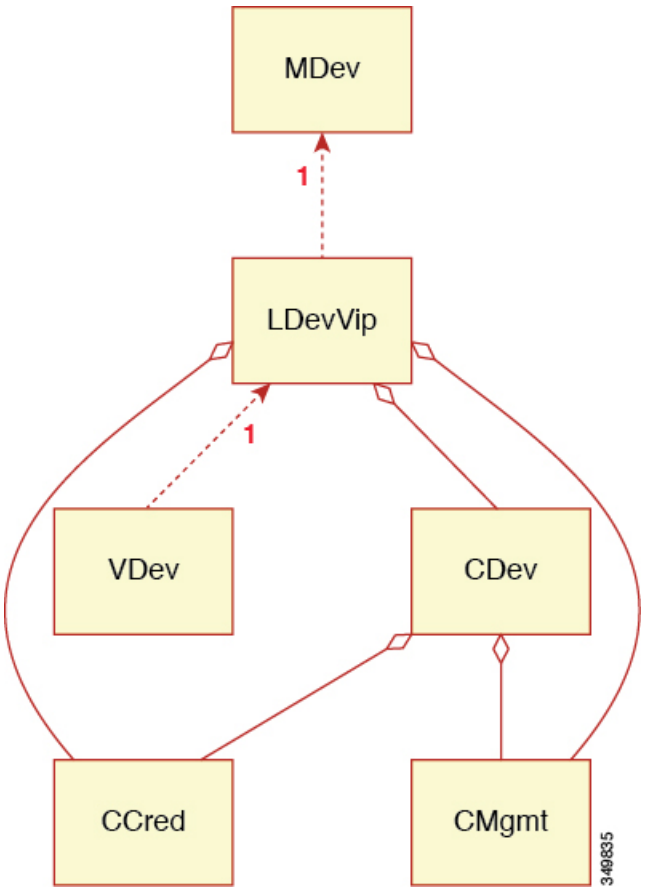


Without a device manager or chassis manager, the model for service devices contains the following key managed objects:

- `MDev`—Represents a device type (vendor, model, version).
- `LDevVIP`—Represents a cluster, a set of identically configured devices for Cold Standby. Contains `CMgmt` and `CCred` for access to the device.
- `CDev`—Represents a member of a cluster, either physical or virtual. Contains `CMgmt` and `CCred` for access to the device.
- `VDev`—Represents a context on a cluster, similar to a virtual machine on a server.

The following figure illustrates the model for the key managed objects, with `CMgmt` (management connectivity) and `CCred` (credentials) included:

Figure 30: Managed Object Model Without a Device Manager or Chassis Manager



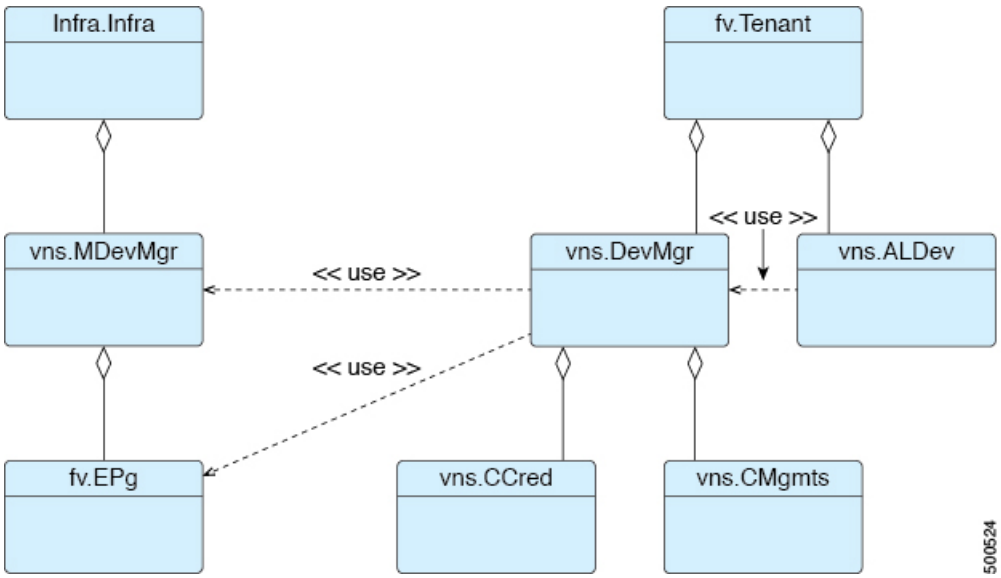
`CMgmt` (host + port) and `CCred` (username + password) allow the script to access the device and cluster.

A device manager and chassis manager adds the ability to control the configuration of clusters and devices from a central management station. The chassis adds a parallel hierarchy to the `MDev` object and `ALDev` object to allow a `CDev` object to be tagged as belonging to a specific chassis. The following managed objects are added to the model to support the device and chassis manager concept:

- `MDevMgr`—Represents a type of device manager. An `MDevMgr` can manage a set of different `MDev`s, which are typically different products from the same vendor.
- `DevMgr`—Represents a device manager. Access to the manager is provided using the contained `CMgmt` and `CCred` managed objects. Each cluster can be associated with only one `DevMgr`.
- `MChassis`—Represents a type of chassis. This managed object is typically included in the package.
- `Chassis`—Represents a chassis instance. It contains the `CMgmt` and `CCred`[Secret] managed objects to provide connectivity to the chassis.

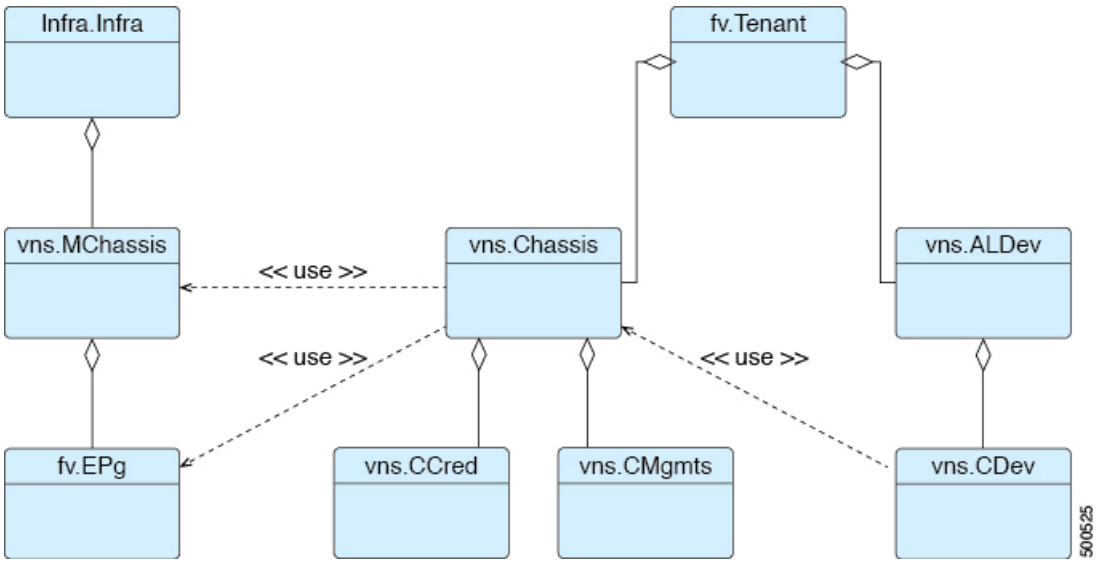
The following figure illustrates the device manager object model:

Figure 31: Device Manager Object Model



The following figure illustrates the chassis manager object model:

Figure 32: Chassis Manager Object Model



Device Manager and Chassis Manager Behavior

The following behavior applies regarding device managers and chassis managers:

- The `DevMgr` object is not required. If there is no relation from `LDevVip` to `DevMgr`, then the system performs callouts without the device manager being defined.

- `Policymgr` performs a sanity check to ensure that the relation from `LDevVip` to `MDev` matches one relation path from `LDevVip` through `DevMgr` and `MDevMgr` to `MDev`. A fault is raised if this is not the case, which prevents further callouts.
- If a relation from `LDevVip` to `DevMgr`, from `DevMgr` to `MDevMgr`, or from `MDevMgr` to the correct `MDev` is added or changed, then the cluster is reset and configured from scratch
- The `Chassis` object is not required. If there is no relation from `CDev` to `Chassis`, then the system performs callouts without the chassis being defined.
- `Policymgr` performs a sanity check to ensure that the relation from `CDev` through `LDevVip` to `MDev` matches one relation path from `CDev` through `Chassis` and `MChassis` to `MDev`. A fault is raised if this is not the case, which prevents further callouts.
- If a relation from `CDev` to `Chassis`, from `Chassis` to `MChassis`, or from `MChassis` to the correct `MDev` is added or changed, then the cluster is reset and configured from scratch

Creating Devices Using the GUI

You can create a device manager for a tenant using the GUI.

Step 1 On the menu bar, choose **Tenants** > **All Tenants**.
Step 2 In the Work pane, double click a tenant's name.
Step 3 In the Navigation pane, choose **Tenant** ***tenant_name*** > **Services** > **L4-L7** > **Device Managers**.
Step 4 In the Work pane, choose **Actions** > **Create Device Manager**.
Step 5 In the **Create Device Manager** dialog box, fill in the fields as required.

Note Policy Based Redirect configuration for Layer 1 or Layer 2 is selected in **Device Manager Type**.

Step 6 Click **Submit**.

Creating a Chassis Using the GUI

You can create a chassis for a tenant using the GUI.

Step 1 On the menu bar, choose **Tenants** > **All Tenants**.
Step 2 In the Work pane, double click a tenant's name.
Step 3 In the Navigation pane, choose **Tenant** ***tenant_name*** > **Services** > **L4-L7** > **Chassis**.
Step 4 In the Work pane, choose **Actions** > **Create Chassis**.
Step 5 In the **Create Chassis** dialog box, fill in the fields as required.
Step 6 Click **Submit**.

Example XML for Device Managers and Chassis Managers

The example XML in the following sections assumes that the `Insieme` package has been loaded, which provides the `uni/infra/mDev-Insieme-Generic-1.0` distinguished name.

Example XML for Creating the MDevMgr Object

The `MDevMgr` object is analogous with the `MDev` object and has vendor, model, and version as the naming properties. Multiple `MDevMgrToMDev` relations can be created if the manager can manage different types of clusters. The following example XML creates the `MDevMgr` object:

```
<polUni>
  <infraInfra>
    <vnsMDevMgr
        vendor="Insieme"
        model="DevMgr"
        version="1.0"
        >
      <vnsRsMDevMgrToMDev tDn="uni/infra/mDev-Insieme-Generic-1.0"/>
    </vnsMDevMgr>
  </infraInfra>
</polUni>
```

The following example XML creates the `MDevMgr` object within the `tenant1` tenant:

```
<polUni>
    <fvTenant name="tenant1">
      <vnsDevMgr name="Foo">
        <vnsCMgmts
            host="10.10.11.11"
            port="1234"/>
        <vnsCMgmts
            host="10.10.11.12"
            port="1234"/>
        <vnsCMgmts
            host="10.10.11.13"
            port="1234"/>
        <vnsCCred name="username" value="admin"/>
        <vnsCCredSecret name="password" value="letmein"/>
        <vnsRsDevMgrToMDevMgr tDn="uni/infra/mDevMgr-Insieme-DevMgr-1.0"/>
    </vnsDevMgr>
  </fvTenant>
</polUni>
```

Example XML for Associating an LDevVip Object With a DevMgr Object

Association of the `LDevVip` object with the `DevMgr` object is done by creating a relation from `LDevVip` to `DevMgr`, as shown in the following example XML:

```
<polUni>
    <fvTenant name="tenant1">
        <vnsLDevVip name="InsiemeCluster" devtype="VIRTUAL">
            ...
            <vnsRsMDevAtt tDn="uni/infra/mDev-Insieme-Generic-1.0"/>
            <vnsRsALDevToDevMgr tDn="uni/tn-tenant1/devMgr-Foo"/>
            ...
        </vnsLDevVip>
```

```
    </fvTenant>
</polUni>
```

Example XML for Creating the MChassis Object

The `MChassis` object is analogous with the `MDev` object and has vendor, model, and version as the naming properties. A `MChassisToMDev` relation determines the device type. The following example XML creates the `MChassis` object:

```
<polUni>
  <infraInfra>
    <vnsMChassis vendor="Insieme" model="DevMgr" version="1.0">
      <vnsRsMChassisToMDev tDn="uni/infra/mDev-Insieme-Generic-1.0"/>
    </vnsMChassis>
  </infraInfra>
</polUni>
```

Example XML for Creating the Chassis Object

The following example XML creates the `Chassis` object:

```
<polUni>
  <fvTenant name="tenant1">
    <vnsChassis name="Foo">
        <vnsCMgmts
            host="10.10.11.11"
            port="1234"/>
        <vnsCMgmts
            host="10.10.11.12"
            port="1234"/>
        <vnsCMgmts
            host="10.10.11.13"
            port="1234"/>
      <vnsCCred name="username" value="admin"/>
      <vnsCCredSecret name="password" value="letmein"/>
      <vnsRsChassisToMChassis tDn="uni/infra/mChassis-Insieme-DevMgr-1.0"/>
    </vnsChassis>
  </fvTenant>
</polUni>
```

Example XML for Associating an CDev Object With a Chassis Object

Association of the `CDev` object with the `Chassis` object is done by creating a relation from `CDev` to `Chassis`, as shown in the following example XML:

```
<polUni>
    <fvTenant name="tenant1">
        <vnsLDevVip name="InsiemeCluster" devtype="VIRTUAL">
           ...
           <vnsCDev name="Generic1" devCtxLbl="C1">
              ...
              <vnsRsCDevToChassis tnVnsChassisName="Foo"/>
            </vnsCDev>
            <vnsRsALDevToDevMgr tDn="uni/tn-coke/devMgr-Foo"/>
           ...
        </vnsLDevVip>
    </fvTenant>
</polUni>
```

Device and Chassis Callouts

The following sections contain device, cluster, and service callout examples that include parameters for the device and chassis manager. The parameters are added to all callouts.

Example deviceValidate Callout for a Device

In the following example `deviceValidate` callout, the device-specific code is shown in bold:

```
2014-10-03 17:38:51,035 DEBUG 140230105585408 [42.42.42.101, 0]:  deviceValidate
{'args': ('1.0',),
 'device': {'creds': {'password': '<hidden>', 'username': 'nsroot'},
            'host': '42.42.42.101',
            'manager': {'creds': {'username': 'admin', 'password': '<hidden>'},
                        'hosts': {'10.10.11.11': {'port': 1234},
                                  '10.10.11.12': {'port': 1234},
                                  '10.10.11.13': {'port': 1234}},
                        'name': 'Foo'},
            'port': 80,
            'version': '1.0',
            'virtual': True}}
```

Example deviceAudit Callout for a Device

In the following example `deviceAudit` callout, the device-specific code is shown in bold:

```
2014-10-03 17:38:56,072 DEBUG 140230088800000 [42.42.42.100, 2]:  deviceAudit
{'args': ({(11, '', 'ext'): {'label': 'in', 'state': 0},
    (11, '', 'int'): {'label': 'out', 'state': 0}},
    {(4, 'oneFolder', 'foo'): {'ackedState': 0,
        'state': 0,
        'transaction': 0,
        'value': {(5, 'oneParam', 'foo'): {'ackedState': 0,
            'state': 0,
            'transaction': 0,
            'value': 'bar'}}}}),
'device': {'creds': {'password': '<hidden>', 'username': 'nsroot'},
    'host': '42.42.42.100',
    'manager': {'creds': {'username': 'admin', 'password': '<hidden>'},
                        'hosts': {'10.10.11.11': {'port': 1234},
                                  '10.10.11.12': {'port': 1234},
                                  '10.10.11.13': {'port': 1234}},
                        'name': 'Foo'},
    'port': 80,
    'version': '1.0',
    'virtual': True}}
```

Example clusterAudit Callout for a Device

In the following example `clusterAudit` callout, the device-specific code is shown in bold:

```
2014-10-03 17:39:01,097 DEBUG 140229734295296 [42.42.42.99, 4]:  clusterAudit
{'args': ({(12, '', 'ext'): {'cifs': {'Generic1': 'ext', 'Generic2': 'ext'},
                             'label': 'in',
                             'state': 0},
           (12, '', 'inside'): {'cifs': {'Generic1': 'ext',
                                         'Generic2': 'ext'},
```

```
                                'label': 'in',
                                'state': 0},
              (12, '', 'int'): {'cifs': {'Generic1': 'int', 'Generic2': 'int'},
                                'label': 'out',
                                'state': 0},
              (12, '', 'outside'): {'cifs': {'Generic1': 'int',
                                             'Generic2': 'int'},
                                    'label': 'out',
                                    'state': 0}},
             {}),
 'device': {'creds': {'password': '<hidden>', 'username': 'nsroot'},
            'devs': {'Generic1': {'creds': {'password': '<hidden>',
                                            'username': 'nsroot'},
                                  'host': '42.42.42.100',
                                  'port': 80,
                                  'virtual': True},
                     'Generic2': {'creds': {'password': '<hidden>',
                                            'username': 'nsroot'},
                                  'host': '42.42.42.101',
                                  'port': 80,
                                  'virtual': True}},
            'host': '42.42.42.99',
            'manager': {'creds': {'username': 'admin', 'password': '<hidden>'},
                        'hosts': {'10.10.10.11': {'port': 1234},
                                  '10.10.10.12': {'port': 1234},
                                  '10.10.10.13': {'port': 1234}},
                        'name': 'Foo'},
            'port': 80,
            'version': '1.0',
            'virtual': True}}
```

Example serviceAudit Callout for a Device

In the following example `serviceAudit` callout, the device-specific code is shown in bold:

```
2014-10-03 17:39:06,169 DEBUG 140229725902592 [42.42.42.99, 5]:  serviceAudit
{'args': ({(0, '', 4474): {'ackedState': 0,
                           'state': 2,
                           'transaction': 0,
                           'txid': 10000,
                           'value': {(1, '', 5787): {
                               ...
                           }},
 'device': {'creds': {'password': '<hidden>', 'username': 'nsroot'},
            'devs': {'Generic1': {'creds': {'password': '<hidden>',
                                            'username': 'nsroot'},
                                  'host': '42.42.42.100',
                                  'port': 80,
                                  'virtual': True},
                     'Generic2': {'creds': {'password': '<hidden>',
                                            'username': 'nsroot'},
                                  'host': '42.42.42.101',
                                  'port': 80,
                                  'virtual': True}},
            'host': '42.42.42.99',
            'manager': {'creds': {'username': 'admin', 'password': '<hidden>'},
                        'hosts': {'10.10.11.11': {'port': 1234},
                                  '10.10.11.12': {'port': 1234},
                                  '10.10.11.13': {'port': 1234}},
                        'name': 'Foo'},
            'port': 80,
            'version': '1.0',
            'virtual': True}}
```

Example deviceValidate Callout for a Chassis

In the following example `deviceValidate` callout, the chassis-specific code is shown in bold:

```
2014-11-13 19:33:16,066 DEBUG 140719921972992 [42.42.42.101, 0]: request: deviceValidate
 {'args': ('1.0',),
  'device': {'chassis': {'creds': {'username': 'admin', 'password': '<hidden>'},
                         'hosts': {'10.10.11.11': {'port': 1234},
                                   '10.10.11.12': {'port': 1234},
                                   '10.10.11.13': {'port': 1234}},
                         'name': 'Foo'},
             'creds': {'username': 'nsroot', 'pasword': '<hidden>'},
             'host': '42.42.42.100',
             'port': 80,
             'virtual': True}}
```

Example deviceAudit Callout for a Chassis

In the following example `deviceAudit` callout, the chassis-specific code is shown in bold:

```
2014-10-03 17:38:56,072 DEBUG 140230088800000 [42.42.42.100, 2]:  deviceAudit
 {'args': ({(11, '', 'ext'): {'label': 'in', 'state': 0},
            (11, '', 'int'): {'label': 'out', 'state': 0}},
           {(4, 'oneFolder', 'one'): {'ackedState': 0,
                         'state': 0,
                         'transaction': 0,
                         'value': {(5, 'oneParam', 'one'): {'ackedState': 0,
                                   'state': 0,
                                   'transaction': 0,
                                   'value': 'foo'}}}}),
  'device': {'chassis': {'creds': {'username': 'admin', 'password': '<hidden>'},
                         'hosts': {'10.10.11.11': {'port': 1234},
                                   '10.10.11.12': {'port': 1234},
                                   '10.10.11.13': {'port': 1234}},
                         'name': 'Foo'},
             'creds': {'password': '<hidden>', 'username': 'nsroot'},
             'host': '42.42.42.101',
             'manager': {'creds': {'username': 'admin', 'password': '<hidden>'},
                         'hosts': {'10.10.10.11': {'port': 1234},
                                   '10.10.10.12': {'port': 1234},
                                   '10.10.10.13': {'port': 1234}},
                         'name': 'Foo'},
             'port': 80,
             'version': '1.0',
             'virtual': True}}
```

Example clusterAudit Callout for a Chassis

In the following example `clusterAudit` callout, the chassis-specific code is shown in bold:

```
2014-10-03 17:39:01,097 DEBUG 140229734295296 [42.42.42.99, 4]:  clusterAudit
 {'args': ({(12, '', 'ext'): {'cifs': {'Generic1': 'ext', 'Generic2': 'ext'},
                              'label': 'in',
                              'state': 0},
            (12, '', 'inside'): {'cifs': {'Generic1': 'ext',
                                          'Generic2': 'ext'},
                                 'label': 'in',
                                 'state': 0},
            (12, '', 'int'): {'cifs': {'Generic1': 'int', 'Generic2': 'int'},
                              'label': 'out',
                              'state': 0},
```

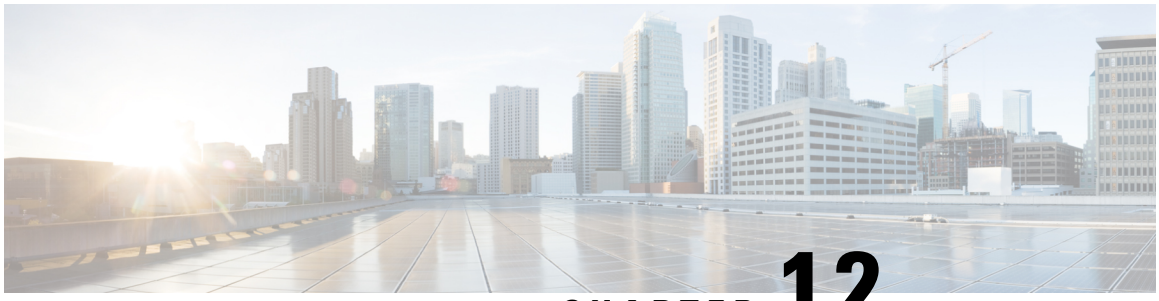
```
          (12, '', 'outside'): {'cifs': {'Generic1': 'int',
                                         'Generic2': 'int'},
                                'label': 'out',
                                'state': 0}},
         {}),
 'device': {'creds': {'password': '<hidden>', 'username': 'nsroot'},
            'devs': {'Generic1': {'chassis': {'creds': {'password': '<hidden>',
                                                        'username': 'admin'},
                                              'hosts': {'10.10.11.11': {'port': 1234},
                                                        '10.10.11.12': {'port': 1234},
                                                        '10.10.11.13': {'port': 1234}},
                                              'name': 'Foo'},
                                  'creds': {'username': 'nsroot', 'pasword': '<hidden>'},
                                  'host': '42.42.42.100',
                                  'port': 80,
                                  'virtual': True},
                     'Generic2': {'chassis': {'creds': {'password': '<hidden>',
                                                        'username': 'admin'},
                                              'hosts': {'10.10.11.11': {'port': 1234},
                                                        '10.10.11.12': {'port': 1234},
                                                        '10.10.11.13': {'port': 1234}},
                                              'name': 'Foo'},
                                  'creds': {'username': 'nsroot', 'pasword': '<hidden>'},
                                  'host': '42.42.42.101',
                                  'port': 80,
                                  'virtual': True}},
            'host': '42.42.42.99',
            'manager': {'creds': {'password': '<hidden>', 'username': 'admin'},
                        'hosts': {'10.10.11.11': {'port': 1234},
                                  '10.10.11.12': {'port': 1234},
                                  '10.10.11.13': {'port': 1234}},
                        'name': 'Foo'},
            'port': 80,
            'version': '1.0',
            'virtual': True}}
```

Example serviceAudit Callout for a Chassis

In the following example `serviceAudit` callout, the chassis-specific code is shown in bold:

```
2014-10-03 17:39:06,169 DEBUG 140229725902592 [42.42.42.99, 5]:  serviceAudit
 {'args': ...,
  'device': {'creds': {'password': '<hidden>', 'username': 'nsroot'},
             'devs': {'Generic1': {'chassis': {'creds': {'username': 'admin',
                                                         'password': '<hidden>'},
                                               'hosts': {'10.10.11.11': {'port': 1234},
                                                         '10.10.11.12': {'port': 1234},
                                                         '10.10.11.13': {'port': 1234}},
                                               'name': 'Foo'},
                                   'creds': {'username': 'nsroot',
                                             'pasword': '<hidden>'},
                                   'host': '42.42.42.100',
                                   'port': 80,
                                   'virtual': True},
                      'Generic2': {'chassis': {'creds': {'username': 'admin',
                                                         'password': '<hidden>'},
                                               'hosts': {'10.10.11.11': {'port': 1234},
                                                         '10.10.11.12': {'port': 1234},
                                                         '10.10.11.13': {'port': 1234}},
                                               'name': 'Foo'},
                                   'creds': {'username': 'nsroot',
                                             'pasword': '<hidden>'},
```

```
                      'host': '42.42.42.101',
                      'port': 80,
                      'virtual': True}},
'host': '42.42.42.99',
'manager': {'creds': {'username': 'admin', 'password': '<hidden>'},
            'hosts': {'10.10.11.11': {'port': 1234},
                      '10.10.11.12': {'port': 1234},
                      '10.10.11.13': {'port': 1234}},
            'name': 'Foo'},
'port': 80,
'version': '1.0',
'virtual': True}}
```



CHAPTER 12

Configuring Unmanaged Mode

- About the Unmanaged Mode, on page 121
- About Managed and Unmanaged Logical Devices, on page 121
- About Managed and Unmanaged Function Nodes, on page 122
- About Layer 4 to Layer 7 Services Endpoint Groups, on page 123
- Using Static Encapsulation for a Graph Connector, on page 123
- Creating a Physical Device Using the NX-OS-Style CLI, on page 123
- Creating a High Availablity Cluster Using the NX-OS-Style CLI, on page 125
- Creating a Virtual Device Using the NX-OS-Style CLI, on page 126
- Example XML for the Unmanaged Mode, on page 127
- Unmanaged Mode Behavior, on page 129

About the Unmanaged Mode

The Layer 4 to Layer 7 service insertion feature enables an administrator to insert one or more services between two endpoint groups. The Application Policy Infrastructure Controller (APIC) allocates the fabric resources (VLANs) for the services and programs fabric (leaf switches) and service appliances as per the configuration specified in the service graph. The APIC requires the device package for a service to be able to use it as part of the service graph. The APIC also programs the service appliance during graph instantiation.

You might want the APIC to allocate only the network resources for the service graph and program only the fabric side during graph instantiation. This might be needed for various reasons, such as if your environment already has an existing orchestrator or a dev-op tool that is more suitable for programming the service appliance. In some other cases, the device package for the service appliance is not available.

The unmanaged mode for services enables you to choose the APIC's behavior for allocating network resources and programming the fabric. When enabled, the unmanaged mode restricts the APIC to allocate only the network resources for a service appliance and to program only the fabric (leaf). The configuration of the device is left to be done externally by you.

About Managed and Unmanaged Logical Devices

The unmanaged mode introduces the `managed` setting to the logical device (`LDevVip`), as shown in the following XML code:

```
<!-- Specified if the device is a managed device-->
<property name="managed"
```

```
        type="scalar:Bool"
        owner="management"
        mod="explicit">
        <default value="true"/>
</property>
```

A device can be either managed or unmanaged. When a device is configured as managed, the Application Policy Infrastructure Controller (APIC) manages the device and programs the device during graph instantiation. By default, when a device is registered with the APIC, the device is set to be in managed mode.

If a device is configured as unmanaged, meaning that the `managed` setting is set to `false`, the APIC does not program the device. The APIC only allocates the network resources and programs the VLAN/VXLAN on the fabric side.

The following settings are not needed when a device cluster is configured as unmanaged:

- Device package
- Connectivity information for the logical device (`vnsLDevViP`) and devices (`CDev`)—management IP address, credentials, and in-band connectivity information
- Information about supported function type (go-through, go-to)
- Information about context awareness (single context or multi-context)

The APIC still needs to know the topology information (`LIF`, `CIF`) for the logical device and devices. This information is needed so that the APIC can program the appropriate ports on the leaf and the APIC can also use this information for troubleshooting wizard purposes.

The APIC also needs to know the relation to `DomP`, which is used for allocating the encapsulation.

About Managed and Unmanaged Function Nodes

The unmanaged mode introduces the `managed` setting to the function node (`AbsNode`), as shown in the following XML code:

```
<!-- Specified if the function is using a managed device-->
<property name="managed"
              type="scalar:Bool"
              owner="management"
              mod="explicit">
              <default value="true"/>
</property>
```

A function node can be either managed or unmanaged. When a function node is configured as managed, the function node can use a managed device. The Application Policy Infrastructure Controller (APIC) programs the device during graph instantiation. By default, when a function node is added to the service graph, the function node is set to be in managed mode.

If a function node is configured as unmanaged, meaning that the `managed` setting is set to `false`, the APIC does not do parameter resolution nor program the devices. The APIC only allocates the network resources and programs the VLAN/VXLAN on fabric side.

The following settings are not needed when a function node is configured as unmanaged:

- `MFunc` relation
- `AbsFuncProfile`
- Configuration parameters (in `AbsNode` or on an endpoint group)
- Information about the supported function type (go-through, go-to)

The APIC still needs to know the network information (`LIF`, `CIF`) for the function node. This information is needed so that the APIC can program the network appropriately on the leaf, and the APIC can also use this information for troubleshooting wizard purposes.

The following settings are still needed:

- `LDevCtx` to enable the selection of `LDevVip` during graph instantiation
- `LIfCtx` to enable the selection of `LIf` during graph instantiation
- Bridge domain in `LIfCtx`
- Route peering in `LIfCtx`
- Subnet in `LIfCtx`

About Layer 4 to Layer 7 Services Endpoint Groups

As part of unmanaged mode feature, the Application Policy Infrastructure Controller (APIC) enables you to specify an endpoint group to be used for the graph connector during graph instantiation. This enables you to better troubleshoot the graph deployment. The APIC uses the Layer 4 to Layer 7 services endpoint group that you specified to download encapsulation information to a leaf. The APIC also uses the endpoint group to create port groups on the distributed virtual switch for virtual devices. A Layer 4 to Layer 7 services endpoint group is also used to aggregate faults and statistics information for a graph connector.

In addition to the enhanced visibility into the deployed graph resources, a Layer 4 to Layer 7 services endpoint group can also be used to specify static encapsulation to be used for a specific graph instance. This encapsulation can also be shared across multiple graph instances by sharing the Layer 4 to Layer 7 services endpoint group across multiple graph instances.

For example XML code that shows how you can use a Layer 4 to Layer 7 services endpoint group with a graph connector, see Example XML of Associating a Layer 4 to Layer 7 Service Endpoint Group with a Connector, on page 128.

Using Static Encapsulation for a Graph Connector

The Application Policy Infrastructure Controller (APIC) allocates the encapsulation for various service graphs during processing. In some use cases, you want to be able explicitly to specify the encapsulation to be used for a specific connector in the service graph. This is known as static encapsulation. Static encapsulations are only supported for the service graph connector that has a service device cluster with physical services. Service device clusters with virtual service devices use the dynamically allocated VLANs from the VMware domain that is associated with the service device cluster.

A static encapsulation can be used with a graph connector by specifying the encapsulation value as part of the Layer 4 to Layer 7 service endpoint group. For example XML code that shows how you can use a static encapsulation with a Layer 4 to Layer 7 services endpoint group, see Example XML of Using Static Encapsulation with a Layer 4 to Layer 7 Service Endpoint Group, on page 128.

Creating a Physical Device Using the NX-OS-Style CLI

This example procedure creates a physical device using the NX-OS-style CLI.

Step 1 Enter the configure mode.

Example:

```
apic1# configure
```

Step 2 Enter the configure mode for a tenant.

```
tenant tenant_name
```

Example:

```
apic1(config)# tenant t1
```

Step 3 Create a cluster:

Example:

```
apic1(config-tenant)# l4l7 cluster name ifav108-asa type physical vlan-domain phyDom5 servicetype FW
```

Step 4 Add a cluster device:

Example:

```
apic1(config-cluster)# cluster-device C1
```

Step 5 Add a provider cluster interface:

Example:

```
apic1(config-cluster)# cluster-interface provider
```

Step 6 Add a member device to the interface:

Example:

```
apic1(config-cluster-interface)# member device C1 device-interface Po1
apic1(config-member)# interface vpc VPCPolASA leaf 103 104
apic1(config-member)# exit
apic1(config-cluster-interface)# exit
```

Step 7 Add a consumer cluster interface:

Example:

```
apic1(config-cluster)# cluster-interface consumer
```

Step 8 Add the same member device to the consumer interface:

Example:

```
apic1(config-cluster-interface)# member device C1 device-interface Po1
apic1(config-member)# interface vpc VPCPolASA leaf 103 104
apic1(config-member)# exit
apic1(config-cluster-interface)# exit
```

Step 9 Exit out of the cluster creation mode:

Example:

```
apic1(config-cluster)# exit
```

Creating a High Availablity Cluster Using the NX-OS-Style CLI

This example procedure creates a high availability cluster using the NX-OS-style CLI.

Step 1 Enter the configure mode.

Example:

```
apic1# configure
```

Step 2 Enter the configure mode for a tenant.

```
tenant tenant_name
```

Example:

```
apic1(config)# tenant t1
```

Step 3 Create a cluster:

Example:

```
apic1(config-tenant)# l4l7 cluster name ifav108-asa type physical vlan-domain phyDom5 servicetype FW
```

Step 4 Add the cluster devices:

Example:

```
apic1(config-cluster)# cluster-device C1
apic1(config-cluster)# cluster-device C2
```

Step 5 Add a provider cluster interface:

Example:

```
apic1(config-cluster)# cluster-interface provider vlan 101
```

Step 6 Add member devices to the interface:

Example:

```
apic1(config-cluster-interface)# member device C1 device-interface Po1
apic1(config-member)# interface vpc VPCPolASA leaf 103 104
apic1(config-member)# exit
apic1(config-cluster-interface)# exit
apic1(config-cluster-interface)# member device C2 device-interface Po2
apic1(config-member)# interface vpc VPCPolASA-2 leaf 103 104
apic1(config-member)# exit
apic1(config-cluster-interface)# exit
```

Step 7 Add another provider cluster interface:

Example:

```
apic1(config-cluster)# cluster-interface provider vlan 102
```

Step 8 Add the same member devices from the first interface to this new interface:

Example:

```
apic1(config-cluster-interface)# member device C1 device-interface Po1
apic1(config-member)# interface vpc VPCPolASA leaf 103 104
apic1(config-member)# exit
```

```
apic1(config-cluster-interface)# exit
apic1(config-cluster-interface)# member device C2 device-interface Po2
apic1(config-member)# interface vpc VPCPolASA-2 leaf 103 104
apic1(config-member)# exit
apic1(config-cluster-interface)# exit
```

Step 9 Exit out of the cluster creation mode:

Example:

```
apic1(config-cluster)# exit
```

Creating a Virtual Device Using the NX-OS-Style CLI

This example procedure creates a virtual device using the NX-OS-style CLI.

Step 1 Enter the configure mode.

Example:

```
apic1# configure
```

Step 2 Enter the configure mode for a tenant.

```
tenant tenant_name
```

Example:

```
apic1(config)# tenant t1
```

Step 3 Create a cluster:

Example:

```
apic1(config-tenant)# l4l7 cluster name ifav108-citrix type virtual vlan-domain ACIVswitch servicetype
 ADC
```

Step 4 Add a cluster device:

Example:

```
apic1(config-cluster)# cluster-device D1 vcenter ifav108-vcenter vm NSVPX-ESX
```

Step 5 Add a consumer cluster interface:

Example:

```
apic1(config-cluster)# cluster-interface consumer
```

Step 6 Add a member device to the consumer interface:

Example:

```
apic1(config-cluster-interface)# member device D1 device-interface 1_1
apic1(config-member)# interface ethernet 1/45 leaf 102
ifav108-apic1(config-member)# vnic "Network adapter 2"
apic1(config-member)# exit
apic1(config-cluster-interface)# exit
```

Step 7 Add a provider cluster interface:

Example:

```
apic1(config-cluster)# cluster-interface provider
```

Step 8 Add the same member device to the provider interface:

Example:

```
apic1(config-cluster-interface)# member device D1 device-interface 1_1
apic1(config-member)# interface ethernet 1/45 leaf 102
ifav108-apic1(config-member)# vnic "Network adapter 2"
apic1(config-member)# exit
apic1(config-cluster-interface)# exit
```

Step 9 Exit out of the cluster creation mode:

Example:

```
apic1(config-cluster)# exit
```

Example XML for the Unmanaged Mode

The example XML in the following sections show how to administer the unmanaged mode.

Example XML of Creating an Unmanaged LDevVip Object

The following example XML creates an unmanaged `LDevVip` object:

```
<polUni>
    <fvTenant name="HA_Tenant1">
        <vnsLDevVip name="ADCCluster1" devtype="VIRTUAL" managed="no">
            <vnsRsALDevToDomP tDn="uni/vmmp-VMware/dom-mininet"/>
        </vnsLDevVip>
    </fvTenant>
</polUni>
```

For Cisco ACI Virtual Edge, the following example XML creates an unmanaged `LDevVip` object associated to the Cisco ACI Virtual Edge VMM domain with `ave` as the switching mode:

```
<polUni>
 <fvTenant name="HA_Tenant1">
  <vnsLDevVip name="ADCCluster1" devtype="VIRTUAL" managed="no">
   <vnsRsALDevToDomP switchingMode="AVE" tDn="uni/vmmp-VMware/dom-mininet_ave"/>
  </vnsLDevVip>
 </fvTenant>
</polUni>
```

Example XML of Creating an Unmanaged AbsNode Object

The following example XML creates an unmanaged `AbsNode` object:

```
    <fvTenant name="HA_Tenant1">
        <vnsAbsGraph name="g1">
            <vnsAbsTermNodeProv name="Input1">
                <vnsAbsTermConn name="C1">
                </vnsAbsTermConn>
            </vnsAbsTermNodeProv>
```

```
            <!-- Node1 provides a service function in un-managed mode -->
            <vnsAbsNode name="Node1" managed="no">
                <vnsAbsFuncConn name="outside" >
                </vnsAbsFuncConn>
                <vnsAbsFuncConn name="inside" >
                </vnsAbsFuncConn>
            </vnsAbsNode>

            <vnsAbsTermNodeCon name="Output1">
                <vnsAbsTermConn name="C6">
                </vnsAbsTermConn>
            </vnsAbsTermNodeCon>

            <vnsAbsConnection name="CON2" >
                <vnsRsAbsConnectionConns
tDn="uni/tn-HA_Tenant1/AbsGraph-g1/AbsTermNodeCon-Output1/AbsTConn"/>
                <vnsRsAbsConnectionConns
tDn="uni/tn-HA_Tenant1/AbsGraph-g1/AbsNode-Node1/AbsFConn-outside"/>
            </vnsAbsConnection>

            <vnsAbsConnection name="CON1" >
                <vnsRsAbsConnectionConns
tDn="uni/tn-HA_Tenant1/AbsGraph-g1/AbsNode-Node1/AbsFConn-inside"/>
                <vnsRsAbsConnectionConns
tDn="uni/tn-HA_Tenant1/AbsGraph-g1/AbsTermNodeProv-Input1/AbsTConn"/>
            </vnsAbsConnection>

        </vnsAbsGraph>
    </fvTenant>
```

Example XML of Associating a Layer 4 to Layer 7 Service Endpoint Group with a Connector

The following example XML associates a Layer 4 to Layer 7 service endpoint group with a connector:

```
<fvTenant name="HA_Tenant1">
   <vnsLDevCtx ctrctNameOrLbl="any" descr="" dn="uni/tn-HA_Tenant1/ldevCtx-c-any-g-any-n-any"

      graphNameOrLbl="any" name="" nodeNameOrLbl="any">
        <vnsRsLDevCtxToLDev tDn="uni/tn-HA_Tenant1/lDevVip-ADCCluster1"/>
        <vnsLIfCtx connNameOrLbl="inside" descr="" name="inside">
            <vnsRsLIfCtxToSvcEPg tDn="uni/tn-HA_Tenant1/ap-sap/SvcEPg-EPG1"/>
            <vnsRsLIfCtxToBD tDn="uni/tn-HA_Tenant1/BD-provBD1"/>
            <vnsRsLIfCtxToLIf tDn="uni/tn-HA_Tenant1/lDevVip-ADCCluster1/lIf-inside"/>
        </vnsLIfCtx>
        <vnsLIfCtx connNameOrLbl="outside" descr="" name="outside">
            <vnsRsLIfCtxToSvcEPg tDn="uni/tn-HA_Tenant1/ap-sap/SvcEPg-EPG2"/>
            <vnsRsLIfCtxToBD tDn="uni/tn-HA_Tenant1/BD-consBD1"/>
            <vnsRsLIfCtxToLIf tDn="uni/tn-HA_Tenant1/lDevVip-ADCCluster1/lIf-outside"/>
        </vnsLIfCtx>
    </vnsLDevCtx>
</fvTenant>
```

Example XML of Using Static Encapsulation with a Layer 4 to Layer 7 Service Endpoint Group

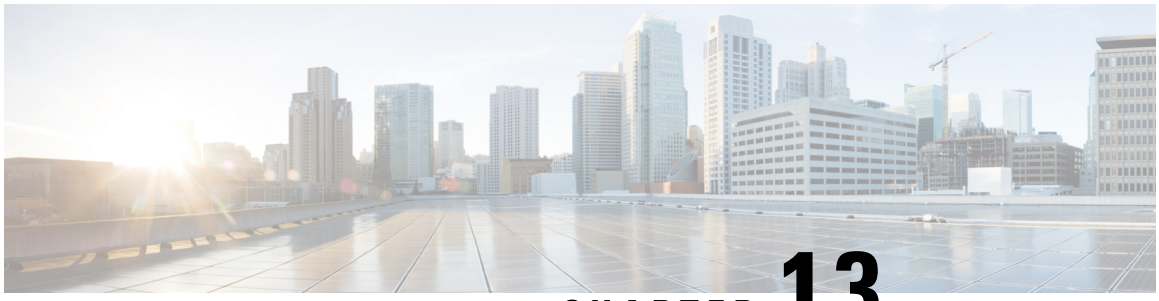
The following example XML uses static encapsulation with a Layer 4 to Layer 7 services endpoint group:

```
<polUni>
    <fvTenant name="HA_Tenant1">
        <fvAp name="sap">
            <vnsSvcEPg name="EPG1" encap="vlan-3510">
            </vnsSvcEPg>
        </fvAp>
    </fvTenant>
</polUni>
```

Unmanaged Mode Behavior

The following behavior applies regarding the unmanaged mode:

- Parameter resolution and unmanaged functions—For an unmanaged function, the Application Policy Infrastructure Controller (APIC) does not perform parameter resolution. Parameters are not required to be configured on `AbsGraph`, an endpoint group, or any other level.
- `VDev` and unmanaged functions— For an unmanaged function, the APIC does not perform parameter resolution or device side programming. No `VDev` tree is created for an unmanaged service graph function.
- Route peering in unmanaged mode—Unmanaged mode does not impact route peering functionality.
- VNIC auto-placement in unmanaged mode—Unmanaged mode does not impact VNIC placement functionality.



CHAPTER 13

Configuring Copy Services

- About Copy Services, on page 131
- Copy Services Limitations, on page 132
- Configuring Copy Services Using the GUI, on page 132
- Configuring Copy Services Using the NX-OS-Style CLI, on page 134
- Configuring Copy Services Using the REST API, on page 136

About Copy Services

Unlike SPAN that duplicates all of the traffic, the Cisco Application Centric Infrastructure (ACI) copy services feature enables selectively copying portions of the traffic between endpoint groups, according to the specifications of the contract. Broadcast, unknown unicast and multicast (BUM), and control plane traffic that are not covered by the contract are not copied. In contrast, SPAN copies everything out of endpoint groups, access ports or uplink ports. Unlike SPAN, copy services do not add headers to the copied traffic. Copy service traffic is managed internally in the switch to minimize impact on normal traffic forwarding.

A copy service is configured as part of a Layer 4 to Layer 7 service graph template that specifies a copy cluster as the destination for the copied traffic. A copy service can tap into different hops within a service graph. For example, a copy service could select traffic between a consumer endpoint group and a firewall provider endpoint group, or between a server load balancer and a firewall. Copy clusters can be shared across tenants.

Copy services require you to do the following tasks:

- Identify the source and destination endpoint groups.
- Configure the contract that specifies what to copy according to the subject and what is allowed in the contract filter.
- Configure Layer 4 to Layer 7 copy devices that identify the target devices and specify the ports where they attach.
- Use the copy service as part of a Layer 4 to Layer 7 service graph template.
- Configure a device selection policy that specifies which device will receive the traffic from the service graph. When you configure the device selection policy, you specify the contract, service graph, copy cluster, and cluster logical interface that is in copy device.

Copy Services Limitations

The following limitations apply when using the copy services feature:

- Copy services are only supported on the N9K-9300-EX and -FX leaf switches.
- For data path traffic that is copied to the local and remote analyzer port, the Class of Service (CoS) and Differentiated Services Code Point (DSCP) values are not preserved in the copied traffic. This is because the contract with the copy action can be hit on either the ingress or egress TOR before or after the actual COS or DSCP value gets modified.

  When policing the data path traffic at a given endpoint ingress direction, the traffic that is copied is the actual incoming traffic before the traffic is policed. This is due to an ASIC limitation in the N9K-93108TC-EX and N9K-93180YC-EX switches.
- Copy services support only one device per copy cluster.
- A copy cluster supports only one logical interface.
- You can configure copy analyzers in the consumer endpoint or provider endpoint only in N9K-93108TC-EX and N9K-93180YC-EX switches. Faults are raised if you configure copy analyzers in N9K-93128TX, N9K-9396PX, or N9K-9396TX switches.
- The `tn-common/ctx-copy` VRF instance, also known as the copy VRF instance, is a system-reserved context for a copy service. The copy VRF instance is auto-configured by the system during the boot up sequence. The copy VRF instance cannot be configured nor deleted by the user.
- Copy services with a vzAny contract is not supported.
- When using a separate copy device for each direction of a flow, you must have two different unidirectional filters.

Configuring Copy Services Using the GUI

This procedure uses the GUI to configure copy services.

Note When you configure a copy device, the context aware parameter is not used. The context aware parameter has a default value of `single context`, which can be ignored.

Step 1 Create one or more copy devices.

For information about creating a copy device, see Creating a Copy Device Using the GUI, on page 133.

Step 2 Create a service graph template to use for copy services.

For information about creating a service graph template, see Configuring a Service Graph Template Using the GUI, on page 38.

a) If you want to create one or more service nodes, drag Layer 4 to Layer 7 service devices from the **Device Clusters** section to in-between the consumer endpoint group and provider endpoint group.

b) Create one or more copy nodes by dragging copy devices from the **Device Clusters** section to in-between any two objects.

The location where you drop the copy device becomes the point in the data flow from where the copy device will copy the traffic.

Step 3 Apply the Layer 4 to Layer 7 service graph template.

For information about applying a service graph template, see Applying a Service Graph Template to Endpoint Groups Using the GUI, on page 41.

Creating a Copy Device Using the GUI

A copy device is used as part of the copy services feature to create a copy node. A copy node specifies at which point of the data flow between endpoint groups to copy traffic.

This procedure only creates a copy device and does not configure anything else that is required to use the copy services feature. For information about configuring copy services, see Configuring Copy Services Using the GUI, on page 132.

Before you begin

You must have configured a tenant.

Step 1 On the menu bar, choose **Tenants** > **All Tenants**.

Step 2 In the Work pane, double-click the tenant's name.

Step 3 In the Navigation pane, choose **Tenant** ***tenant_name*** > **Services** > **L4-L7** > **Devices**.

Step 4 In the Work pane, choose **Actions** > **Create Copy Devices**.

Step 5 In the **Create Copy Devices** dialog box, in the **General** section, complete the following fields:

Name	Description
Name field	Enter a name for the copy device.
Device Type buttons	The device type. A copy device can only be a physical device.
Physical Domain drop-down list	Choose the physical domain for the device.

Step 6 In the **Device 1** section, click + to add a device interface, complete the following fields, and then click **Update**:

Name	Description
Name field	Enter a name for the device interface.
Path drop-down list	Choose a port, port channel, or virtual port channel for the device interface to use. The copy device connects to that port, port channel, or virtual port channel and copies traffic from it.

Step 7 In the **Cluster** section, click + to add a cluster interface, complete the following fields, and then click **Update**:

Name	Description
Name field	Enter a name for the cluster interface.
Concrete Interfaces drop-down list	Choose one or more concrete interfaces for the cluster interface to use.

Encap field	Enter a VLAN to use for encapsulation. The VLAN name format is as follows:
	`vlan-#`
	# is the VLAN's ID. For example:
	`vlan-12`

Step 8 Click **Submit**.

Configuring Copy Services Using the NX-OS-Style CLI

This procedure provides examples of using the CLI to configure copy services.

Note When you configure a copy device, the context aware parameter is not used. The context aware parameter has a default value of `single context`, which can be ignored.

Step 1 Create a copy cluster.

Example:

```
l4l7 cluster name Copy_1 type physical vlan-domain phys_scale_copy service COPY function none
  cluster-device Copy_1_Device_1
  cluster-interface Tap_copy vlan 3644
    member device Copy_1_Device_1 device-interface int1
      interface ethernet 1/15 leaf 104
      exit
    member device Copy_1_Device_1 device-interface int2
      interface ethernet 1/15 leaf 105
      exit
    member device Copy_1_Device_1 device-interface int3
      interface ethernet 1/20 leaf 105
      exit
    exit
  exit
```

Step 2 Create an abstract graph and device context, and then apply the graph.

Example:

```
l4l7 graph g5 contract c5
  service CP1 device-cluster-tenant t1 device-cluster Copy_1 mode OTHER service COPY
    connector copy cluster-interface Tap_copy
      exit
    exit
  connection C1 terminal consumer terminal provider copyservice CP1 connector copy
  Exit
```

Step 3 Attach the contract to the graph.

Example:

```
contract c5
  scope tenant
  subject Subject
    access-group default both
    l4l7 graph g5
    exit
  Exit
```

Step 4 Attach the endpoint groups to the contract.

Example:

```
epg epg2210
  bridge-domain member bd5
  contract consumer c5
  exit
epg epg2211
  bridge-domain member bd5
  contract provider c5
  Exit
```

Example

The following example creates a firewall service graph with a copy device on both sides:

```
tenant tenant_cmd_line
  l4l7 graph graph_fire contract fire
    service Fire device-cluster-tenant tenant_cmd_line device-cluster Fire mode FW_ROUTED

      connector consumer cluster-interface Outside_cmdline
        bridge-domain tenant tenant_cmd_line name Consumer_BD_1
        exit
      connector provider cluster-interface Inside_cmdline
        bridge-domain tenant tenant_cmd_line name Provider_BD1
        exit
      exit
    service CP2 device-cluster-tenant tenant_cmd_line device-cluster copy1 mode OTHER
     service COPY
      connector copy cluster-interface int1
        exit
      exit
    service CP3 device-cluster-tenant tenant_cmd_line device-cluster copy1 mode OTHER
     service COPY
      connector copy cluster-interface int1
        exit
      exit
    connection C1 terminal consumer service Fire connector consumer copyservice CP2
     connector copy
    connection C2 terminal provider service Fire connector provider copyservice CP3
     connector copy
    exit
  Exit
```

The following example creates a firewall and load balance in one-arm mode with copy devices attached in all the links:

```
l4l7 graph Graph_LB_Firewall contract c1_firewall
  service Fire device-cluster-tenant Tenant_Firewall_LB device-cluster Firewall_1 mode
```

```
  FW_ROUTED
   connector consumer cluster-interface Outside_Firewall
     bridge-domain tenant Tenant_Firewall_LB name BD1_Consumer
     exit
   connector provider cluster-interface Inside_Firewall
     bridge-domain tenant Tenant_Firewall_LB name BD2_Provider
     exit
   exit
service LB device-cluster-tenant Tenant_Firewall_LB device-cluster LB_1 mode ADC_ONE_ARM

   connector consumer cluster-interface LB_Inside
     bridge-domain tenant Tenant_Firewall_LB name BD2_Provider
     exit
   connector provider cluster-interface LB_Inside
     bridge-domain tenant Tenant_Firewall_LB name BD2_Provider
     exit
   Exit
 service CP6 device-cluster-tenant Tenant_Pass2 device-cluster Copy_pass2 mode OTHER
  service-type COPY
   connector copy cluster-interface tap_copy
     exit
   Exit
 service CP7 device-cluster-tenant Tenant_Pass2 device-cluster Copy_pass2 mode OTHER
  service-type COPY
   connector copy cluster-interface tap_copy
     exit
   Exit
 service CP8 device-cluster-tenant Tenant_Pass2 device-cluster Copy_pass2 mode OTHER
  service-type COPY
   connector copy cluster-interface tap_copy
     exit
   exit
 connection C1 terminal consumer service Fire connector consumer copyservice CP6
  connector copy
 connection C2 intra-service service1 Fire connector1 provider service2 LB connector2
  consumer copyservice CP7 connector copy
 connection C3 terminal provider service LB connector provider copyservice CP8
  connector copy
 exit
exit
```

Configuring Copy Services Using the REST API

A copy device is used as part of the copy services feature to create a copy node. A copy node specifies at which point of the data flow between endpoint groups to copy traffic.

This procedure provides examples of using the REST API to configure copy services.

Note When you configure a copy device, the context aware parameter is not used. The context aware parameter has a default value of `single context`, which can be ignored.

Before you begin

You must have configured a tenant.

Step 1 Create a copy device.

Example:

```
<vnsLDevVip contextAware="single-Context" devtype="PHYSICAL" funcType="None" isCopy="yes"
  managed="no" mode="legacy-Mode" name="copy0" packageModel="" svcType="COPY" trunking="no">
    <vnsRsALDevToPhysDomP tDn="uni/phys-phys_scale_copy"/>
    <vnsCDev devCtxLbl="" name="copy_Dyn_Device_0" vcenterName="" vmName="">
        <vnsCIf name="int1" vnicName="">
            <vnsRsCIfPathAtt tDn="topology/pod-1/paths-104/pathep-[eth1/15]"/>
        </vnsCIf>
        <vnsCIf name="int2" vnicName="">
            <vnsRsCIfPathAtt tDn="topology/pod-1/paths-105/pathep-[eth1/15]"/>
        </vnsCIf>
    </vnsCDev>
    <vnsLIf encap="vlan-3540" name="TAP">
        <vnsRsCIfAttN tDn="uni/tn-t22/lDevVip-copy0/cDev-copy_Dyn_Device_0/cIf-[int2]"/>
        <vnsRsCIfAttN tDn="uni/tn-t22/lDevVip-copy0/cDev-copy_Dyn_Device_0/cIf-[int1]"/>
    </vnsLIf>
</vnsLDevVip>
```

Step 2 Create a logical device context (also known as a device selection policy).

Example:

```
<vnsLDevCtx ctrctNameOrLbl="c0" descr="" graphNameOrLbl="g0" name="" nodeNameOrLbl="CP1">
    <vnsRsLDevCtxToLDev tDn="uni/tn-t22/lDevVip-copy0"/>
    <vnsLIfCtx connNameOrLbl="copy" descr="" name="">
        <vnsRsLIfCtxToLIf tDn="uni/tn-t22/lDevVip-copy0/lIf-TAP"/>
    </vnsLIfCtx>
</vnsLDevCtx>
```

Step 3 Create and apply the copy graph template.

Example:

```
<vnsAbsGraph descr="" name="g0" ownerKey="" ownerTag="" uiTemplateType="UNSPECIFIED">
    <vnsAbsTermNodeCon descr="" name="T1" ownerKey="" ownerTag="">
        <vnsAbsTermConn attNotify="no" descr="" name="1" ownerKey="" ownerTag=""/>
        <vnsInTerm descr="" name=""/>
        <vnsOutTerm descr="" name=""/>
    </vnsAbsTermNodeCon>
    <vnsAbsTermNodeProv descr="" name="T2" ownerKey="" ownerTag="">
        <vnsAbsTermConn attNotify="no" descr="" name="1" ownerKey="" ownerTag=""/>
        <vnsInTerm descr="" name=""/>
        <vnsOutTerm descr="" name=""/>
    </vnsAbsTermNodeProv>
    <vnsAbsConnection adjType="L2" connDir="provider" connType="external" descr="" name="C1"
      ownerKey="" ownerTag="" unicastRoute="yes">
        <vnsRsAbsConnectionConns tDn="uni/tn-t22/AbsGraph-g0/AbsTermNodeCon-T1/AbsTConn"/>
        <vnsRsAbsConnectionConns tDn="uni/tn-t22/AbsGraph-g0/AbsTermNodeProv-T2/AbsTConn"/>
        <vnsRsAbsCopyConnection tDn="uni/tn-t22/AbsGraph-g0/AbsNode-CP1/AbsFConn-copy"/>
    </vnsAbsConnection>
    <vnsAbsNode descr="" funcTemplateType="OTHER" funcType="None" isCopy="yes" managed="no"
      name="CP1" ownerKey="" ownerTag="" routingMode="unspecified" sequenceNumber="0"
      shareEncap="no">
        <vnsAbsFuncConn attNotify="no" descr="" name="copy" ownerKey="" ownerTag=""/>
        <vnsRsNodeToLDev tDn="uni/tn-t22/lDevVip-copy0"/>
    </vnsAbsNode>
</vnsAbsGraph>
```

Step 4 Define the relation to the copy graph in the contract that is associated with the endpoint groups.

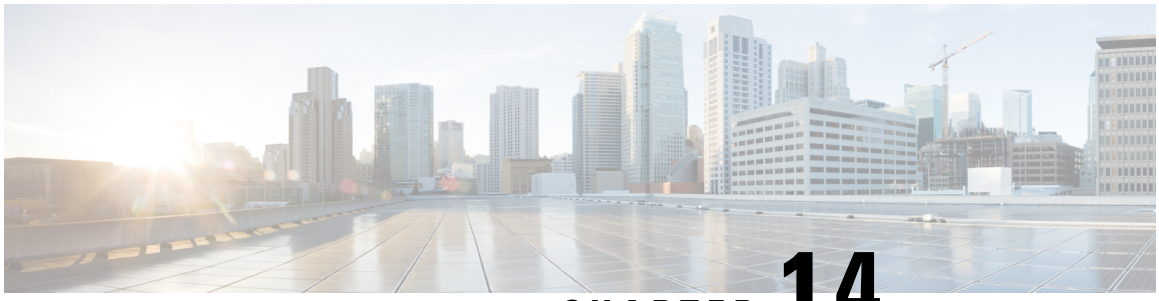
Example:

```
<vzBrCP descr="" name="c0" ownerKey="" ownerTag="" prio="unspecified" scope="tenant"
  targetDscp="unspecified">
    <vzSubj consMatchT="AtleastOne" descr="" name="Subject" prio="unspecified"
      provMatchT="AtleastOne" revFltPorts="yes" targetDscp="unspecified">
        <vzRsSubjFiltAtt directives="" tnVzFilterName="default"/>
        <vzRsSubjGraphAtt directives="" tnVnsAbsGraphName="g0"/>
    </vzSubj>
</vzBrCP>
```

Step 5 Attach the contract to the endpoint group.

Example:

```
<fvAEPg name="epg2860">
    <fvRsCons tnVzBrCPName="c0"/>
    <fvRsBd tnFvBDName="bd0"/>
    <fvRsDomAtt tDn="uni/phys-phys_scale_SB"/>
    <fvRsPathAtt tDn="topology/pod-1/paths-104/pathep-[PC_int2_g1]" encap="vlan-2860"
      instrImedcy="immediate"/>
</fvAEPg>
<fvAEPg name="epg2861">
    <fvRsProv tnVzBrCPName="c0"/>
    <fvRsBd tnFvBDName="bd0"/>
    <fvRsDomAtt tDn="uni/phys-phys_scale_SB"/>
    <fvRsPathAtt tDn="topology/pod-1/paths-105/pathep-[PC_policy]" encap="vlan-2861"
      instrImedcy="immediate"/>
</fvAEPg>
```



CHAPTER 14

Configuring Layer 4 to Layer 7 Resource Pools

- About Layer 4 to Layer 7 Resource Pools, on page 139
- About External and Public IP Address Pools, on page 139
- About External Layer 3 Routed Domains and the Associated VLAN Pools, on page 140
- About External Routed Networks, on page 140
- Supported Managed Mode Layer 4 to Layer 7 Devices, on page 140
- About Cloud Orchestrator Mode Function Profiles, on page 141
- Creating an IP Address Pool for Layer 4 to Layer 7 Resource Pools Using the GUI, on page 141
- Creating a Dynamic VLAN Pool for Layer4 to Layer 7 Resource Pools Using the GUI, on page 142
- Creating an External Routed Domain for Layer 4 to Layer 7 Resource Pools Using the GUI, on page 142
- Preparing Layer 4 to Layer 7 Devices for Use in Layer 4 to Layer 7 Resource Pools, on page 143
- Validating the APIC Configuration of a Layer 4 to Layer 7 Device for Use in a Layer 4 to Layer 7 Resource Pool, on page 143
- Configuring the Device Management Network and Routes, on page 144
- Creating a Layer 4 to Layer 7 Resource Pool, on page 144
- Configuring a Layer 4 to Layer 7 Resource Pool Using the GUI, on page 146

About Layer 4 to Layer 7 Resource Pools

Layer 4 to Layer 7 resource pools bring together related configurations with regard to deploying Layer 4 to Layer 7 service devices. The related configuration is packaged together so that it can be used by orchestration layers such as Cisco Application Centric Infrastructure (Cisco ACI) Windows Azure Pack integration to deploy Layer 4 to Layer 7 service devices.

About External and Public IP Address Pools

For Layer 4 to Layer 7 resource pools created in Cisco APIC Release 3.0(x) and earlier, the public and external IP address pools were one and the same and were simply marked as external. For Layer 4 to Layer 7 resource pools created in Cisco APIC Release 3.1(x) and later, there is a separation and distinction between these two types of address pools. External IP address pools are used for the external interface of the Layer 4 to Layer 7 device, and L3Out SVI IP allocation. For Layer 4 to Layer 7 devices that are connected through a VPC into the fabric, 3 IP addresses are consumed by the L3Out configuration (side A primary IP address, side B primary IP address, and secondary IP address) while port channel and single interface connections consume 2 IP addresses (primary IP address and secondary IP address).

Public IP address pools are used to allocate dynamic NAT IP addresses (1 per tenant VRF), load balancers, virtual IP addresses (1 per tenant EPG), and additional public NAT IP addresses.

By separating the two IP address types, a Cisco APIC administrator is able to achieve the following:

- Export only the IP addresses in the IP pool marked as public - hiding the device-level interface IP addresses
- Incrementally add to the public IP address pool's varying blocks of IP addresses as they are acquired and available to the common tenant L3Out

About External Layer 3 Routed Domains and the Associated VLAN Pools

The external L3Out routed domain is used to provision the L3Out for both the internal and external connectors of the Layer 4 to Layer 7 devices. These L3Outs allow for traffic to originate from outside of the Cisco Application Centric Infrastructure (Cisco ACI) fabric and be able to reach the resources that are inside of the Cisco ACI fabric. The L3Outs also allow for traffic to originate from within the Cisco ACI fabric and be able to reach outside of the Cisco ACI fabric. The VLANs within the VLAN pool that are associated with the Layer 3 routed domain must be unique for a given leaf or VPC leaf switch pair where the Layer 4 to Layer 7 service devices are connected. If the Layer 4 to Layer 7 service devices span across multiple leaf or VPC leaf switch pairs, then the limitation also extends to these leaf and VPC leaf switch pairs.

Note VLAN blocks should not be reconfigured or removed from the VLAN pools once the Layer 4 to Layer 7 resource pool is in use. You can add VLAN blocks to the current VLAN block if required for expansion.

The following VLAN pool sizing considerations apply:

- 1 VLAN is dynamically allocated per external IP address pool
- 1 VLAN is dynamically allocated per tenant virtual forwarding and routing (VRF) that is accessing the Layer 4 to Layer 7 resource pool
- The external routed domain and the associated VLAN pool can be used across Layer 4 to Layer 7 resource pools

About External Routed Networks

For information about configuring external routed networks, see the *Cisco APIC Layer 3 Outside for Tenant Networks* document at the following URL:

http://www.cisco.com/c/en/us/support/cloud-systems-management/application-policy-infrastructure-controller-apic/tsd-products-support-series-home.html

Supported Managed Mode Layer 4 to Layer 7 Devices

Layer 4 to Layer 7 resource pools currently support the following managed mode Layer 4 to Layer 7 devices:

Resource Pool Type	Device Type	Device Package Version	Device Firmware Version	Device Models	Service Type	Context Aware
Classic	Physical/Virtual	CISCO-ASA-1.2 (1.2.7.10) or newer	9.2.1 or newer	ASA55xx/ASAv	Firewall	No
Classic	Physical/Virtual	Citrix-Netscaler-1.0 (11.0 Build 65.36) or newer	11.1 Build 49.16 or newer	Netscaler MPX/VPX/SDX Contexts	ADC	No
Cloud Orchestrator Mode	Physical/Virtual	CISCO-ASA-1.3 (1.3.10.8) or newer	9.6 or newer	ASA55xx/ASAv	Firewall	No
Cloud Orchestrator Mode	Physical/Virtual	Citrix-Netscaler-2.0 or newer	12.0 or newer	Netscaler MPX/VPX/SDX Contexts	ADC	No

About Cloud Orchestrator Mode Function Profiles

Cloud orchestrator mode function profiles are function profiles within Layer 4 to Layer 7 device packages that allow for a simplified and consistent configuration of standard services. In the case of an ADC Layer 4 to Layer 7 device providing load balancer functionality, a single load balancer policy can be pushed to an endpoint group. This load balancer policy would not need to change when different ADC Layer 4 to Layer 7 devices are selected, simplifying the administrator's configuration overhead. Layer 4 to Layer 7 device packages are provided by their respective device vendors.

Creating an IP Address Pool for Layer 4 to Layer 7 Resource Pools Using the GUI

The following procedure creates an IP address pool for Layer 4 to Layer 7 resource pools using either GUI mode.

Step 1 On the menu bar, choose **Tenants** > **Common**.

Step 2 In the **Navigation** pane, choose **Tenant Common** > **IP Address Pools**.

Step 3 In the **Work** pane, choose **Actions** > **Create IP Address Pool**.

Step 4 In the **Create IP Address Pool** dialog box, fill in the fields as required.

Do not include the gateway address in the **Address Ranges**. The gateway address will be used as the secondary IP address of the Layer 4 to Layer 7 device external L3Out, which will act as a pervasive gateway.

Example:

- **Name**—`ExtIPPool1`
- **Gateway Address**— `132.121.101.1/24`
- **Address Block**

- **From**—`132.121.101.2`
- **To**—`132.121.101.200`

Step 5 Click **Submit**.

Creating a Dynamic VLAN Pool for Layer4 to Layer 7 Resource Pools Using the GUI

The following procedure creates a dynamic VLAN pool for Layer 4 to Layer 7 resource pools using the GUI mode.

Step 1 On the menu bar, choose **Fabric** > **Access Policies**.

Step 2 In the **Navigation** pane, choose **Pools** > **VLAN**.

Step 3 In the **Work** pane, choose **Actions** > **Create VLAN Pool**.

Step 4 In the **Create VLAN Pool** dialog box, fill in the fields as required, except as specified below:

a) For the **Allocation Mode** buttons, click **Dynamic Allocation**.

b) On the **Encap Blocks** table, click **+**.

c) In the **Create Ranges** dialog box, fill in the fields as specified below:

- In the **Range** fields, enter the desired VLAN range.
- For the **Allocation Mode** buttons, click **Inherit alloc mode from parent**.

d) Click **OK**.

Step 5 In the **Create VLAN Pool** dialog box, click **Submit**.

Creating an External Routed Domain for Layer 4 to Layer 7 Resource Pools Using the GUI

The following procedure creates a dynamic VLAN pool for Layer 4 to Layer 7 resource pools using the GUI mode.

Step 1 On the menu bar, choose **Fabric** > **Access Policies**.

Step 2 In the **Navigation** pane, choose **Physical and External Domains** > **External Routed Domains**.

Step 3 In the **Work** pane, choose **Actions** > **Create Layer 3 Domain**.

Step 4 In the **Create Layer 3 Domain** dialog box, fill in the fields as required, except as specified below:

a) For the **Associated Attachable Entity Profile** drop-down list, choose the attachable entity profile to which all of the Layer 4 to Layer 7 service devices are connected.

b) For the **VLAN Pool** drop-down list, choose the dynamic VLAN pool that you created for Layer 4 to Layer 7 resource pools.
c) On the **Security Domains** table, add any required security domains.

Step 5 Click **Submit**.

Preparing Layer 4 to Layer 7 Devices for Use in Layer 4 to Layer 7 Resource Pools

To configure the physical connectivity of the Layer 4 to Layer 7 devices, see the appropriate configuration guide for each respective device regarding port channel or VPC configuration within the device.

Note For ASA55xx firewall devices that are context aware, the path configuration must be consistent across all the ASA contexts for a given physical ASA55xx. Configuring ASA contexts using different interfaces is not allowed in this configuration.

Validating the APIC Configuration of a Layer 4 to Layer 7 Device for Use in a Layer 4 to Layer 7 Resource Pool

The following procedure validates the Cisco Application Policy Infrastructure Controller (Cisco APIC) configuration of a Layer 4 to Layer 7 services device for use in Layer 4 to Layer 7 resource pools using the GUI mode.

Step 1 On the menu bar, choose **Tenants** > **Common**.

Step 2 In the Navigation pane, choose **Tenant** ***tenant_name*** > **Services** > **L4-L7** > **Devices** > ***ASA_or_NetScaler_logical_device_name*** > ***concrete_device_name***.

Step 3 In the **Work** pane, choose the **Policy** tab.

Step 4 In the **Interfaces** table, verify that there are at least 2 interfaces, with each one mapping to a validate path (port, port channel, or vPC) in the fabric.

Step 5 For each ASA or NetScaler, verify that there is both a **Cluster > consumer** interface and a **Cluster > provider** interface defined. Even if the NetScalers will be used for internal load balancing, having such a configuration allows the tenant to use the NetScaler in both private and public IP address load balancing.

Step 6 For HA configurations, verify that there are 2 concrete interfaces for each cluster interface. Doing so will ensure that each port, port channel, or vPC will be configured correctly.

Step 7 For each device type, configure the necessary onboarding configuration parameters, such as enabling load balancing on the NetScaler through the Layer 4 to Layer 7 parameters of the device in tenant common.

Configuring the Device Management Network and Routes

You must configure the management routes and remove the default route out of band directly on the Layer 4 to Layer 7 device.

The following example uses the Cisco Application Policy Infrastructure Controller (Cisco APIC) NX-OS-style CLI to configure the management route of an ASA firewall:

```
apic1(config)# route management 10.24.24.0 255.255.255.0 172.0.0.1
```

The following example uses the Cisco APIC NX-OS-style CLI to remove the default route:

```
apic1(config)# no route 0.0.0.0 0.0.0.0 172.0.0.1
```

The following example uses the Citrix NetScaler CLI to configure the management route of a NetScaler Application Delivery Controller (ADC) load balancer:

```
> add route 10.24.24.0 255.255.255.0 172.0.0.1
```

The following example uses the Citrix NetScaler CLI to remove the default route:

```
> rm route 0.0.0.0 0.0.0.0 172.0.0.1
```

Creating a Layer 4 to Layer 7 Resource Pool

Creating a Layer 4 to Layer 7 Resource Pool Using the GUI

The following procedure creates a Layer 4 to Layer 7 resource pool using the GUI mode. Once the resource pool has allocated various components for use by the tenants, you cannot modify to the resource pool. You can perform maintenance tasks such as adding IP address blocks, adding VLAN blocks, and adding logical devices, such as an ASA firewall or Citrix NetScaler load balancer.

Step 1 On the menu bar, choose **Tenants** > **Common**.

Step 2 In the **Navigation** pane, choose **Tenant Common** > **Services** > **L4-L7** > **L4-L7 Resource Pools**.

Step 3 In the **Work** pane, choose **Actions** > **Create L4-L7 Resource Pool**.

Step 4 In the **Create L4-L7 Resource Pool** dialog box, fill in the fields as required, except as specified below:

a) In the **Private IP Address Subnet** field, enter the subnet that is used for internal device interface IP addresses, internal VIP addresses, and internal L3Out IP addresses.

b) For the **External IP Address Pool** drop-down list, choose the IP address pool that is used for the dynamic allocation of IP addresses used throughout the service graph and devices. You can create a new IP address pool if necessary. For **Connect Type**, choose **L3 External Network**.

c) For the **Public IP Address Pool** table, choose the IP address pool that is used for the dynamic allocation of IP addresses used for NAT IP addressing and VIP addressing. You can create a new IP address pool, if necessary. For **Connect Type**, choose **L3 External Network**.

d) For the **External Routed Domain** drop-down list, choose the external routed domain that you created for use in this Layer 4 to Layer 7 resource pool. You can create a new external routed domain if necessary.

e) In the **External Routed Networks** table, add the external routed networks that the tenants can consume.

The first external routed network will automatically be marked as `Default`. Only the default routed network is currently used.

f) In the **L4-L7 Devices** table, add the Layer 4 to Layer 7 devices that will be part of this Layer 4 to Layer 7 resource pool.
g) In the **Functional Profiles** table, add the cloud orchestrator mode profiles associated with the Layer 4 to Layer 7 devices you added. For example, if you have added a Citrix Netscaler ADC L4-L7 Device, choose the Netscaler Device Package option in the **Device Package** drop-down menu then choose a cloud orchestrator mode-enabled profile in the **Function Profile** drop-down menu.

Step 5 Click **Submit**.

Creating a Layer 4 to Layer 7 Resource Pool Using the NX-OS-Style CLI

This section provides example commands for using the NX-OS-style CLI to configure Layer 4 to Layer 7 resource pools.

Step 1 Enter the configure mode.

```
apic1# configure
```

Step 2 Enter the configure mode for tenant common.

```
apic1(config)# tenant common
```

Step 3 Specify the Layer 4 to Layer 7 resource pool.

```
apic1(config)# l4l7 resource-pool <resource pool name>
```

Step 4 Set the resource pool version.

```
apic1(config-resource-pool)# version normalized
```

Note The version can be:

- **classic**–For resource pools created before Cisco APIC Release 3.1(x).
- **normalized**–For resource pools created after Cisco APIC Release 3.1(x). Supports devices and device packages with cloud orchestrator mode.

Step 5 Associate an Layer 4 to Layer 7 devices to a resource pool.

```
apic1(config-resource-pool)# l4l7-cluster Dev-ASA-4
apic1(config-resource-pool)# l4l7-cluster Dev-MPX-4
```

Step 6 Associate an IP address pool as an external IP address pool to a resource pool.

```
apic1(config-resource-pool)# address-pool mininetExtPoolL3Ext l3-external
```

Step 7 (For normalized resource pools) Associate an IP address pool as a public IP address pool to a resource pool.

```
apic1(config-resource-pool)# public-address-pool mininetPubPoolL3Ext l3-external
```

Step 8 Associate to an external routed domain.

```
apic1(config-resource-pool)# external-routed-domain L3ServicesDom
```

Step 9 Configure the private IP address subnet for a resource pool.

```
apic1(config-resource-pool)# subnet 192.168.254.1/24
```

Step 10 Associate to an L3Out EPG in tenant common.

```
apic1(config-resource-pool)# l3out vpcDefaultInstP default
```

Step 11 (Optional, for normalized resource pools) Associate cloud orchestrator mode function profiles to the resource pool.

```
apic1(config-resource-pool)# function-profile ASAContainer
apic1(config-function-profile)# device-package CISCO-ASA-1.2
apic1(config-function-profile)# service-function-profile
CISCO-ASA-1.2/WebServiceProfileGroup/WebPolicyForRoutedModeCloud
apic1(config-function-profile)# exit
apic1(config-resource-pool)#
```

Configuring a Layer 4 to Layer 7 Resource Pool Using the GUI

Configuring Layer 4 to Layer 7 Devices in a Resource Pool

Adding Layer 4 to Layer 7 Devices to a Layer 4 to Layer 7 Resouce Pool

Note A dedicated VLAN will be consumed for each for each L3Out created for the tenant in their private VRF. The dynamic VLAN pool associated with the Layer 3 domain may need additional VLANs added to accommodate the additional devices to the resource pool.

You can add new Layer 4 to Layer 7 devices to the resource pool at any time.

Step 1 On the menu bar, choose **Tenants** > **Common**.

Step 2 In the **Navigation** pane, choose **Tenant Common** > **Services** > **L4-L7** > **L4-L7 Resource Pools**.

The resource pools appear in the **Navigation** pane as a drop-down list under **L4-L7 Resource Pools**.

Step 3 Click the Layer 4 to Layer 7 resource pool to which you want to add a device.

Step 4 From the Work pane, click the **L4-L7 Devices** tab.

Step 5 From the **L4-L7 Devices** table, click the plus icon (+).

The **Create An L4-L7 Device** dialog appears.

Step 6 Click the **Device** drop-down arrow and choose a Layer 4 to Layer 7 device.

Step 7 Click **Submit**.

Removing Layer 4 to Layer 7 Devices from a Layer 4 to Layer 7 Resource Pool

The resource pool is unusable by any tenants without available Layer 4 to Layer 7 devices configured. If the L4-L7 Device is not allocated and exported to any tenants, perform the following:

Step 1 On the menu bar, choose **Tenants** > **Common**.

Step 2 In the **Navigation** pane, choose **Tenant Common** > **Services** > **L4-L7** > **L4-L7 Resource Pools**.

The resource pools appear in the **Navigation** pane as a drop-down list under **L4-L7 Resource Pools**.

Step 3 Click the Layer 4 to Layer 7 resource pool with the device you want to remove.

Step 4 From the Work pane, click the **L4-L7 Devices** tab.

Step 5 Click to highlight the Layer 4 to Layer 7 device you want to remove then click the **trashcan** icon.

A confirmation dialog appears.

Step 6 Click **Yes** to confirm the deletion.

Configuring External IP Address Pools in a Resource Pool

Adding an External IP Address Pool to a Layer 4 to Layer 7 Resource Pool

If the resource pool is in use, do not remove or update the external IP address pool as it is in use by tenants.

Step 1 On the menu bar, choose **Tenants** > **Common**.

Step 2 In the **Navigation** pane, choose **Tenant Common** > **Services** > **L4-L7** > **L4-L7 Resource Pools**.

The resource pools appear in the **Navigation** pane as a drop-down list under **L4-L7 Resource Pools**.

Step 3 Click the Layer 4 to Layer 7 resource pool to which you want to add an external IP address pool.

Step 4 From the Work pane, click the **Basic** tab.

Step 5 From the **External IP Address Pool** table, click the plus icon (+) .

The **External IP Address Pool** fields appear.

Step 6 Click the **Connect Type** drop-down arrow and choose **L3 External Network** then enter the appropriate values in the remaining **External IP Address Pool** fields.

Note For a description of a field, click the help icon (**?**) in the top-right corner.

Step 7 Click **Update**.

Removing an External IP Address Pool from a Layer 4 to Layer 7 Resource Pool

Note

- If the resource pool is in use, do not remove or update the external IP address pool as it is in use by tenants.
- If removing, adding, or updating the external IP address pool to handle IP address pool exhaustion, do not remove and add a larger IP address pool. In these situations, create a new Layer 4 to Layer 7 resource pool with a similar configuration such as Layer 3 domain and an L3Out but with a new external IP address pool.
- The resource pool is unusable by any tenants without an external IP address pool configured.

Step 1 On the menu bar, choose **Tenants** > **Common**.

Step 2 In the **Navigation** pane, choose **Tenant Common** > **Services** > **L4-L7** > **L4-L7 Resource Pools**.

The resource pools appear in the **Navigation** pane as a drop-down list under **L4-L7 Resource Pools**.

Step 3 Click the Layer 4 to Layer 7 resource pool with the external IP address pool you want to remove.

Step 4 From the Work pane, click the **Basic** tab.

Step 5 From the **External IP Address Pool** table, click to highlight the external IP address pool you want to remove then click the **trashcan** icon.

A confirmation dialog appears.

Step 6 Click **Yes** to confirm the deletion.

Configuring Public IP Address Pools in a Resource Pool

Adding Public IP Address Pools to a Layer 4 to Layer 7 Resource Pool

Note

- For Layer 4 to Layer 7 resource pools created in Cisco APIC Release 3.0(x) and earlier, the external IP address pool is used as the public IP address pool and should not be modified once in use by any tenants.
- For Layer 4 to Layer 7 resource pools created in Cisco APIC Release 3.1(x) and later, you can add new public IP address pools to the resource pool at any time.
- The resource pool is unusable by any tenant without public IP address pools configured.

Step 1 On the menu bar, choose **Tenants** > **Common**.

Step 2 In the **Navigation** pane, choose **Tenant Common** > **Services** > **L4-L7** > **L4-L7 Resource Pools**.

The resource pools appear in the **Navigation** pane as a drop-down list under **L4-L7 Resource Pools**.

Step 3 Click the Layer 4 to Layer 7 resource pool to which you want to add a public IP address pool.

Step 4 From the Work pane, click the **Basic** tab.

Step 5 From the **Public IP Address Pool** table, click the plus icon (**+**).

The **Public IP Address Pool** fields appear.

Step 6 Click the **Connect Type** drop-down arrow and choose **L3 External Network** then enter the appropriate values in the remaining **External IP Address Pool** fields.

Note For a description of a field, click the help icon (**?**) in the top-right corner.

Step 7 Click **Update**.

Removing Public IP Address Pools from a Layer 4 to Layer 7 Resource Pool

Note

- For Layer 4 to Layer 7 resource pools created in Cisco APIC Release 3.0(x) and earlier, the external IP address pool is used as the public IP address pool and should not be modified once in use by any tenants.
- For Layer 4 to Layer 7 resource pools created in Cisco APIC Release 3.1(x) and later, removing IP address pools from the resource pool should not be performed if any tenants are currently utilizing the IP address pool.
- The resource pool is unusable by any tenant if no public IP address pools configured.

Step 1 On the menu bar, choose **Tenants** > **Common**.

Step 2 In the **Navigation** pane, choose **Tenant Common** > **Services** > **L4-L7** > **L4-L7 Resource Pools**.

The resource pools appear in the **Navigation** pane as a drop-down list under **L4-L7 Resource Pools**.

Step 3 Click the Layer 4 to Layer 7 resource pool with the public IP address pool you want to remove.

Step 4 From the Work pane, click the **Basic** tab.

Step 5 From the **Public IP Address Pool** table, click to highlight the public IP address pool you want to remove then click the **trashcan** icon.

A confirmation dialog appears.

Step 6 Click **Yes** to confirm the deletion.

Updating an External Routed Domain for a Layer 4 to Layer 7 Resource Pool

The resource pool is unusable by any tenant if no external routed domain is configured.

Step 1 On the menu bar, choose **Tenants** > **Common**.

Step 2 In the **Navigation** pane, choose **Tenant Common** > **Services** > **L4-L7** > **L4-L7 Resource Pools**.

The resource pools appear in the **Navigation** pane as a drop-down list under **L4-L7 Resource Pools**.

Step 3 Click the Layer 4 to Layer 7 resource pool with the external routed domain you want to update.

Step 4 From the Work pane, click the **External** tab.

Step 5 Click the **External Routed Domain** drop-down arrow and choose a Layer 3 domain.

Step 6 Click **Submit**.

Updating External Routed Networks for a Layer 4 to Layer 7 Resource Pool

The resource pool is unusable by any tenant if no external routed networks are configured.

Step 1 On the menu bar, choose **Tenants** > **Common**.

Step 2 In the **Navigation** pane, choose **Tenant Common** > **Services** > **L4-L7** > **L4-L7 Resource Pools**.

The resource pools appear in the **Navigation** pane as a drop-down list under **L4-L7 Resource Pools**.

Step 3 Click the Layer 4 to Layer 7 resource pool with the external routed network you want to update.

Step 4 From the Work pane, click the **External** tab.

Step 5 From the **External Routed Networks** table, click the plus icon (**+**).

The **External Routed Networks** fields appear.

Step 6 Enter the appropriate value in the **External Routed Networks** fields.

Note For a description of a field, click the help icon (**?**) in the top-right corner.

Step 7 Click **Update**.

Configuring Cloud Orchestrator Mode Function Profiles in a Resource Pool

Adding a Cloud Orchestrator Mode Function Profile to a Layer 4 to Layer 7 Resource Pool

Note

- For Layer 4 to Layer 7 resource pools created in Cisco APIC Release 3.0(x) and earlier, the cloud orchestrator mode function profile is not available. A default function profile is used for the Layer 4 to Layer 7 device.
- For Layer 4 to Layer 7 resource pools created in Cisco APIC Release 3.1(x) and later, there should be only 1 cloud orchestrator mode function profile for a given Layer 4 to Layer 7 device package in the resouce pool at any given time.
- Function profiles in the Layer 4 to Layer 7 resource pools are used only for the initial service graph creation. If the function profiles are updated in the resource pool, only newly deployed graphs will receive that configuration. If no function profiles are defined in the resource pool, a default function profile is used for Cisco ASA and Citrix Netscaler Layer 4 to Layer 7 devices.

Step 1 On the menu bar, choose **Tenants** > **Common**.

Step 2 In the **Navigation** pane, choose **Tenant Common** > **Services** > **L4-L7** > **L4-L7 Resource Pools**.

The resource pools appear in the **Navigation** pane as a drop-down list under **L4-L7 Resource Pools**.

Step 3 Click the Layer 4 to Layer 7 resource pool to which you want to add a cloud orchestrator mode function profile.

Step 4 From the Work pane, click the **Function Profiles** tab.

Step 5 From the **Function Profiles** table, click the plus icon (**+**).

The **Create a Function Profile** dialog appears.

Step 6 Enter the appropriate values in the fields.

Note For a description of a field, click the help icon (**?**) in the top-right corner.

a) Enter a name in the **Name** field.
b) Click the **Device Package** drop-down arrow and choose the device package that has a cloud orchestrator mode function profile.
c) Click the **Function Profile** drop-down arrow and choose a cloud orchestrator mode function profile.

Step 7 Click **Submit**.

Removing a Cloud Orchestrator Mode Function Profile from a Layer 4 to Layer 7 Resource Pool

Note

- For Layer 4 to Layer 7 resource pools created in Cisco APIC Release 3.0(x) and earlier, the cloud orchestrator mode function profile is not available. A default function profile is used for the Layer 4 to Layer 7 device.
- For Layer 4 to Layer 7 resource pools created in Cisco APIC Release 3.1(x) and later, there should be only 1 cloud orchestrator mode function profile for a given Layer 4 to Layer 7 device package in the resouce pool at any given time.
- The resource pool is unusable by any tenant unless a cloud orchestrator mode function profile is configured for the device in the pool.

Step 1 On the menu bar, choose **Tenants** > **Common**.

Step 2 In the **Navigation** pane, choose **Tenant Common** > **Services** > **L4-L7** > **L4-L7 Resource Pools**.

The resource pools appear in the **Navigation** pane as a drop-down list under **L4-L7 Resource Pools**.

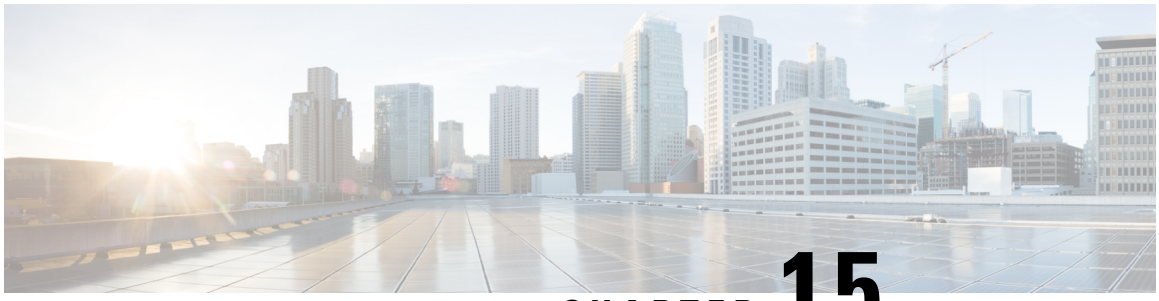
Step 3 Click the Layer 4 to Layer 7 resource pool to which you want to add a cloud orchestrator mode function profile.

Step 4 From the Work pane, click the **Function Profiles** tab.

Step 5 From the **Function Profiles** table, click to highlight the function profile you want to remove then click the **trashcan** icon.

A confirmation dialog appears.

Step 6 Click **Yes** to confirm the deletion.



CHAPTER 15

Configuration Parameters

- Configuration Parameters Inside the Device Package Specification, on page 153
- Configuration Parameters Inside An Abstract Function Profile, on page 156
- Configuration Parameters Inside an Abstract Function Node in a Service Graph, on page 160
- Configuration Parameters Inside Various Configuration MOs, on page 163
- Parameter Resolution, on page 166
- Looking Up an MO During Parameter Resolution, on page 167
- About Role-Based Access Control Rule Enhancements, on page 168

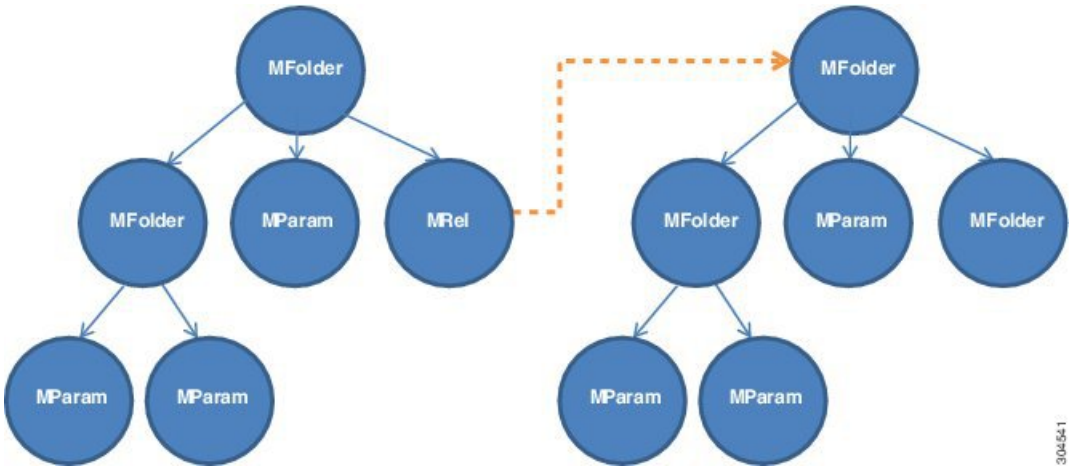
Configuration Parameters Inside the Device Package Specification

A device package contains an XML file that describes the specification for the service device. This specification includes device information as well as various functions provided by the service device.

As part of the device specification, this file must contain the declaration for the configuration needed by the service device. This configuration is needed to configure various functions that are provided by the service device during graph instantiation.

The following figure shows the configuration parameters hierarchy inside the device package.

Figure 33: Configuration Parameters Hierarchy Inside the Device Package



MFolder

MFolder is a group of configuration items that can contain MParams and other nested MFolders. An MFolder has following attributes:

Attribute	Description
Key	Defines the type of the configuration item. The key is defined in the device package and can never be overwritten. The key is used as a matching criterion as well as for validation.
Description	Describes the configuration item.
Cardinality	Specifies the cardinality of the configuration item. The default value of cardinality is `1`. If cardinality is `N`, The Application Policy Infrastructure Controller (APIC) allows N instances of the configuration parameter to be configured.
ScopedBy	Specifies the scope for the parameter resolution. ScopedBy determines where to look for parameter values when APIC resolves the parameter from configuration MOs. Default value is `Epg`. Supported values are `Tenant`, `Ap`, `Bd`, and `Epg`.
RsConnector	A relation that associates a configuration item to an MConn.
DevCtx	Allows a configuration item to be associated with a specific physical device (CDev) in a device (LDev).
Locked	Allows a configuration item value to be locked. Once locked, the value cannot be changed.

MParam

MParam is the basic unit of configuration parameters that declares a single configuration parameter. MParam has following attributes:

Attribute	Description
Key	Defines the type of the configuration item. The key is defined in the device package and can never be overwritten. The key is used as a matching criterion as well as for validation.
Description	Describes the configuration item.

Attribute	Description
Cardinality	Specifies the cardinality of the configuration item. The default value of cardinality is `1`. If cardinality is `N`, The APIC allows N instances of the configuration parameter to be configured.
RsConnector	A relation that associates a configuration item to an MConn.
Mandatory	Allows a configuration item to be marked as mandatory.
Locked	Allows a configuration item value to be locked. Once locked, the value cannot be changed.
Validation	Specifies the validation method for the value.

MRel

MRel allows one MFolder to refer to another MFolder. Using MRel inside an MFolder, an administrator can associate the containing MFolder to the MFolder that is pointed by the MRel using the RsTarget relation contained inside MRel. MRel has following attributes:

Attribute	Description
Key	Defines the type of the configuration item. The key is defined in the device package and can never be overwritten. The key is used as a matching criterion as well as for validation.
Description	Describes the configuration item.
Cardinality	Specifies the cardinality of the configuration item. The default value of cardinality is `1`.
RsTarget	A relation that associates a configuration folder to another MFolder. The value of TDn for this relation is the DN of the target folder.
RsConnector	A relation that associates a configuration item to an MConn.
Mandatory	Allows a configuration item to be marked as mandatory.

Configuration Scope of a Device Package Specification

In a device specification file, configuration items are arranged in different sections.

MDevCfg

The MDevCfg section describes the device level configuration, which is shared by all service graphs using the device. The Application Policy Infrastructure Controller (APIC) does a reference counting of the configuration objects created by using the configuration items described in this section. Objects are only deleted from a service device after all the graph instances that use the device are deleted.

MFuncCfg

The MFuncCfg describes the configuration that is local to a service function and is specific to a service function. The APIC does a reference counting of the configuration objects created by the configuration items described in this section. Objects are created and deleted whenever a service function is instantiated and deleted.

MGrpCfg

The MGrpCfg describes the configuration that is shared by all functions of a service graph using the device. The APIC does a reference counting of the configuration objects created by using the configuration items described in this section. Objects are only deleted from a service device after all functions from the service graph are deleted.

Example XML of Configuration Parameters Inside the Device Package

The following XML example shows configuration parameters inside of the device package:

```
<vnsMFolder key="VServer" scopedBy="epg">
    <vnsRsConnector tDn="uni/infra/mDev-Acme-ADC-1.0/mFunc-SLB/mConn-external"/>
    <vnsMParam key="vservername" description="Name of VServer" mandatory="true"/>
    <vnsMParam key="vip" description="Virtual IP"/>
    <vnsMParam key="subnet" description="Subnet IP"/>
    <vnsMParam key="port" description="Port for Virtual server"/>
    <vnsMParam key="persistencetype" description="persistencetype"/>
    <vnsMParam key="servicename" description="Service bound to this vServer"/>
    <vnsMParam key="servicetype" description="Service bound to this vServer"/>
    <vnsMParam key="clttimeout" description="Client timeout"/>
    <vnsMFolder key="VServerGlobalConfig"
      description="This references the global configuration">
        <vnsMRel key="ServiceConfig">
            <vnsRsTarget tDn="uni/infra/mDev-Acme-ADC-1.0/mDevCfg/mFolder-Service"/>
        </vnsMRel>
        <vnsMRel key="ServerConfig">
            <vnsRsTarget tDn="uni/infra/mDev-Acme-ADC-1.0/mDevCfg/mFolder-Server"/>
        </vnsMRel>
        <vnsMRel key="VipConfig">
            <vnsRsTarget
              tDn="uni/infra/mDev-Acme-ADC-1.0/mDevCfg/mFolder-Network/mFolder-vip"/>
            <vnsRsConnector tDn="uni/infra/mDev-Acme-ADC-1.0/mFunc-SLB/mConn-external"/>
        </vnsMRel>
    </vnsMFolder>
</vnsMFolder>
```

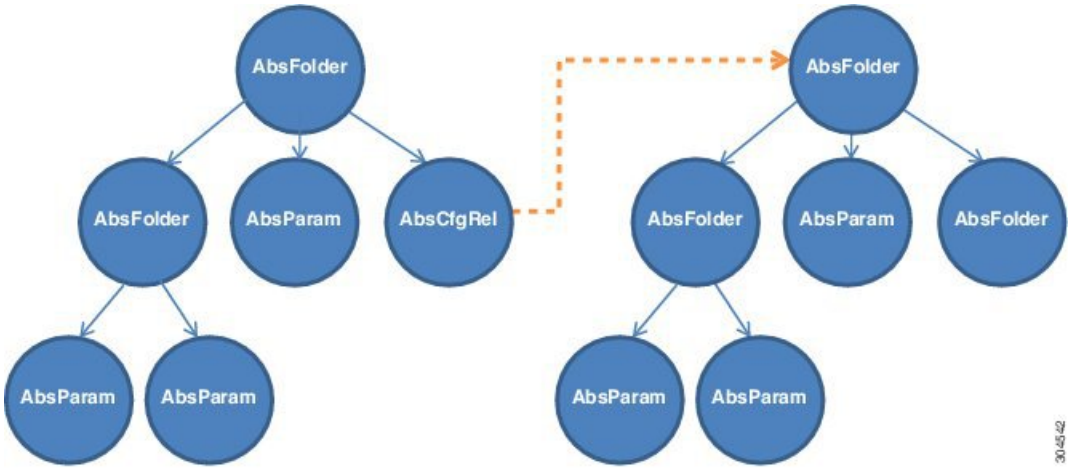
Configuration Parameters Inside An Abstract Function Profile

An abstract profile allows an administrator to configure the default values for the configuration parameters. An abstract function profile contains configuration parameters with values. These values are used as default values during graph instantiation.

An abstract function profile is attached to a function node in a service graph. The default values specified in an abstract function profile is then used when rendering the function onto the service device at the graph instantiation time.

The following figure shows the configuration parameters hierarchy within an abstract function profile.

Figure 34: Configuration Parameters Hierarchy Within an Abstract Function Profile



AbsFolder

AbsFolder is a group of configuration items that can contain AbsParams and other nested AbsFolders. For each AbsFolder, there must be an MFolder inside the device package. The Application Policy Infrastructure Controller (APIC) validates each AbsFolder to ensure that the AbsFolder has a corresponding MFolder in the package. An AbsFolder has the following attributes:

Attribute	Description
Key	Defines the type of the configuration item. The key is defined in the device package and can never be overwritten. The key is used as a matching criterion as well as for validation.
Description	Describes the configuration item.
Cardinality	Specifies the cardinality of the configuration item. The default value is `1`.
ScopedBy	Specifies the scope for the parameter resolution. ScopedBy determines where to look for parameter values when the APIC resolves the parameter from configuration MOs. Default value is `Epg`. Supported values are `Tenant`, `Ap`, `Bd`, and `Epg`.
DevCtx	Allows a configuration item to be associated with a specific physical device (CDev) in a device cluster (LDev).
Locked	Allows a configuration item value to be locked. Once locked, the value cannot be changed.

AbsParam

AbsParam is the basic unit of configuration parameters. AbsParam defines a single configuration parameter. As with an AbsFolder, each AbsParam must have an equivalent MFolder in the device specification. The APIC validates the specification to ensure that AbsParam has a corresponding MFolder in the package. The value of an AbsParam is validated using the validation method specified in MParam. An AbsParam has following attributes:

Attribute	Description
Key	Defines the type of the configuration item. The key is defined in the device package and can never be overwritten. The key is used as a matching criterion as well as for validation.
Value	Holds the value for a given configuration item. Values are not supported for MParam.
Description	Describes the configuration item.
Cardinality	Specifies the cardinality of the configuration item. The default value is `1`.
Mandatory	Allows a configuration item to be marked as mandatory.
Locked	Allows a configuration item value to be locked. Once locked, the value cannot be changed.
Validation	Specifies the validation mechanism to be used to validate the configuration parameter.

AbsRel

AbsRel allows one AbsFolder to refer to another AbsFolder. An AbsRel has following attributes:

Attribute	Description
Key	Defines the type of the configuration item. The key is defined in the device package and can never be overwritten. The key is used as a matching criterion as well as for validation.
Value	Holds the value for a given configuration item. Values are not supported for MParam.
Description	Describes the configuration item.
Cardinality	Specifies the cardinality of the configuration item. The default value is `1`.
Mandatory	Allows a configuration item to be marked as mandatory.

Configuration Scope of an Abstract Function Profile

Within an abstract function profile, the configuration parameters are structured in a similar manner as to how they are structured inside a device package. There are three different scopes.

AbsDevCfg

This section provides the default value for the configuration items that are declared to be the device level configuration inside of a device package. The configuration items are specified under MDevCfg.

For each configuration item, there must be an equivalent configuration item under the device package.

The configuration that is described in this section is shared by all service graphs that use the device. The Application Policy Infrastructure Controller (APIC) does a reference counting of the configuration objects that are created by using the configuration items that are described in this section. Objects are only deleted from a service device after all graph instances that are using the device are deleted.

AbsGrpCfg

This section provides the default value for the configuration items that are declared to be the device level configuration inside of a device package. The configuration items are specified under MGrpCfg.

For each configuration item, there must be an equivalent configuration item under the device package.

The configuration described in this section is shared by all functions of a service graph that use the device. The APIC does a reference counting of the configuration objects that are created by using the configuration items that are described in this section. Objects are only deleted from a service device after all graph instances that are using the device are deleted.

AbsFuncCfg

This section provides the default value for the configuration items that are declared to be the function level configuration inside of a device package. The configuration items are specified under MFuncCfg.

For each configuration item, there must be an equivalent configuration item under the device package.

This section is used to describe the configuration that is local to a service function. The configuration described in this section is specific to a service function. The APIC does a reference counting of the configuration objects that are created by the configuration items that are described in this section. Objects are created and deleted whenever a service function is instantiated and deleted.

Example XML POST for an Abstract Function Profile With Configuration Parameters

The following XML POST example shows an abstract function profile with configuration parameters:

```
<vnsAbsFuncProfContr name = "NP">
    <vnsAbsFuncProfGrp name = "Grp1">
        <vnsAbsFuncProf name = "P1">
            <vnsRsProfToMFunc tDn="uni/infra/mDev-Acme-ADC-1.0/mFunc-SLB"/>
            <vnsAbsDevCfg name="D1">
                <vnsAbsFolder key="Service" name="Service-Default" cardinality="n">
                    <vnsAbsParam name="servicetype" key="servicetype" value="TCP"/>
                    <vnsAbsParam name="serviceport" key="serviceport" value="80"/>
                    <vnsAbsParam name="maxclient" key="maxclient" value="1000"/>
                    <vnsAbsParam name="maxreq" key="maxreq" value="100"/>
                    <vnsAbsParam name="cip" key="cip" value="enable"/>
                    <vnsAbsParam name="usip" key="usip" value="enable"/>
                    <vnsAbsParam name="sp" key="sp" value=""/>
                    <vnsAbsParam name="svrtimeout" key="svrtimeout" value="60"/>
                    <vnsAbsParam name="clttimeout" key="clttimeout" value="60"/>
                    <vnsAbsParam name="cka" key="cka" value="NO"/>
                    <vnsAbsParam name="tcpb" key="tcpb" value="NO"/>
                    <vnsAbsParam name="cmp" key="cmp" value="NO"/>
                </vnsAbsFolder>
            </vnsAbsDevCfg>
            <vnsAbsFuncCfg name="SLB">
                <vnsAbsFolder key="VServer" name="VServer-Default">
                    <vnsAbsParam name="port" key="port" value="80"/>
                    <vnsAbsParam name="persistencetype" key="persistencetype"
                      value="cookie"/>
                    <vnsAbsParam name="clttimeout" key="clttimeout" value="100"/>
                    <vnsAbsParam name="servicetype" key="servicetype" value="TCP"/>
                    <vnsAbsParam name="servicename" key="servicename"/>
                </vnsAbsFolder>
            </vnsAbsFuncCfg>
        </vnsAbsFuncProf>
    </vnsAbsFuncProfGrp>
</vnsAbsFuncProfContr>
```

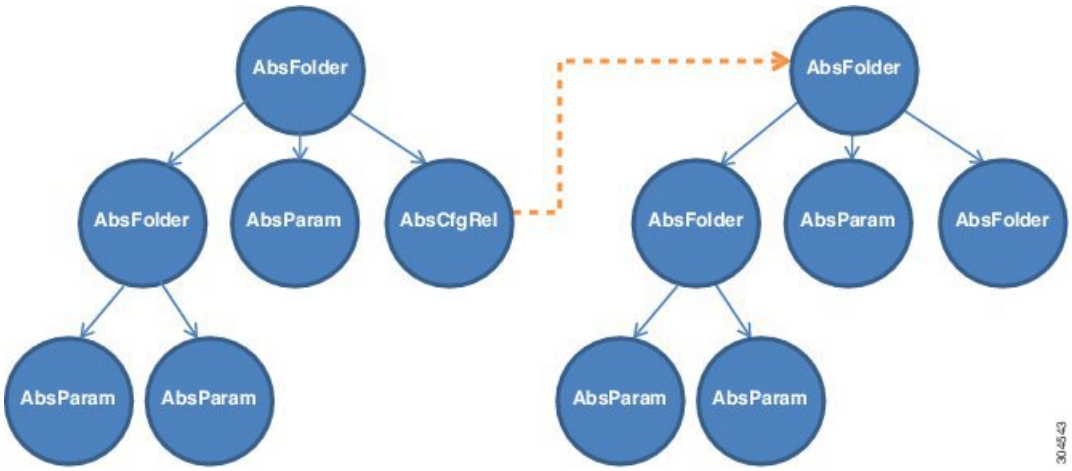
Configuration Parameters Inside an Abstract Function Node in a Service Graph

A function node inside a service graph allows an administrator to configure values for the configuration parameters. These values are used during graph instantiation.

Within an abstract function node, the configuration parameters are structured in a similar manner as to how they are structured inside an abstract function profile.

The following figure shows the configuration parameters hierarchy inside an abstract function node.

Figure 35: Configuration Parameters Hierarchy Inside an Abstract Function Node



AbsDevCfg

This section is used to provide the default value for the configuration items that are declared to be the device level configuration inside of the device package. The configuration items are specified under MDevCfg.

For each of these configuration items, there must be an equivalent configuration item under the device package.

AbsGrpCfg

This section is used to provide the default value for the configuration items that are declared to be the device level configuration inside of the device package. The configuration items are specified under MGrpCfg.

For each of these configuration items, there must be an equivalent configuration item under the device package.

The configuration that is described in this section is shared by all functions of a service graph that use the device. The Application Policy Infrastructure Controller (APIC) does a reference counting of the configuration objects that are created by using the configuration items that are described in this section. Objects are only deleted from a service device after all functions from the service graph are deleted.

AbsFuncCfg

This section is used to provide the default value for the configuration items that are declared to be the function level configuration inside of the device package. The configuration items are specified under MFuncCfg.

For each of these configuration items, there must be an equivalent configuration item under the device package.

This section is used to describe the configuration that is local to a service function. The configuration that is described in this section is specific to a service function. The APIC does a reference counting of the configuration objects that are created by the configuration items that are described in this section. Objects are created and deleted whenever a service function is instantiated and deleted.

AbsFolder

AbsFolder is a group of configuration items that can contain AbsParamss and other nested AbsFolders. For each AbsFolder, there must be an MFolder inside the device package. The APIC validates each AbsFolder to ensure that the AbsFolder has a corresponding MFolder in the package. AbsFolder has the following attributes:

Attribute	Description
Key	Defines the type of the configuration item. The key is defined in the device package and can never be overwritten. The key is used as a matching criterion as well as for validation.
Description	Describes the configuration item.
Cardinality	Specifies the cardinality of the configuration item. The default value is `1`.
ScopedBy	Specifies the scope for the parameter resolution. ScopedBy determines where to look for parameter values when the APIC resolves the parameter from configuration MOs. Default value is `Epg`. Supported values are `Tenant`, `Ap`, `Bd`, and `Epg`.
RsCfgToConn	A relation that associates a configuration item to an AbsConn.
DevCtx	Allows a configuration item to be associated with a specific physical device (CDev) in a device (LDev).
Locked	Allows a configuration item value to be locked. Once locked, the value cannot be changed.

AbsParam

AbsParam is the basic unit of configuration parameters. AbsParam defines a single configuration parameter. As with an AbsFolder, each AbsParam must have an equivalent MFolder in the device specification. The APIC validates the specification to ensure that AbsParam has a corresponding MFolder in the package. The value of an AbsParam is validated using the validation method specified in MParam. AbsParam has following attributes:

Attribute	Description
Key	Defines the type of the configuration item. The key is defined in the device package and can never be overwritten. The key is used as a matching criterion as well as for validation.
Value	Holds the value for a given configuration item. Values are not supported for MParam.
Description	Describes the configuration item.
Cardinality	Specifies the cardinality of the configuration item. The default value is `1`.
RsCfgToConn	A relation that associates a configuration item to an MConn.

Attribute	Description
Mandatory	Allows a configuration item to be marked as mandatory.
Locked	Allows a configuration item value to be locked. Once locked, the value cannot be changed.
Validation	Specifies the validation mechanism to be used to validate the configuration parameter.

AbsRel

AbsRel allows one AbsFolder to refer to another AbsFolder. AbsRel has following attributes:

Attribute	Description
Key	Defines the type of the configuration item. The key is defined in the device package and can never be overwritten. The key is used as a matching criterion as well as for validation.
Value	Holds the value for a given configuration item. Values are not supported for MParam.
Description	Describes the configuration item.
Cardinality	Specifies the cardinality of the configuration item. The default value is `1`.
RsCfgToConn	A relation that associates a configuration item to an MConn.
Mandatory	Allows a configuration item to be marked as mandatory.
Locked	Allows a configuration item value to be locked. Once locked, the value cannot be changed.

Example XML POST for an Abstract Function Node With Configuration Parameters

The following XML POST example shows an abstract function node with configuration parameters:

```
<vnsAbsNode name = "SLB" funcType="GoTo" >
    <vnsRsDefaultScopeToTerm tDn="uni/tn-tenant1/AbsGraph-G3/AbsTermNode-Output1/outtmnl"/>

     <vnsAbsFuncConn name = "C4" direction = "input">
         <vnsRsMConnAtt tDn="uni/infra/mDev-Acme-ADC-1.0/mFunc-SLB/mConn-external" />
     </vnsAbsFuncConn>
     <vnsAbsFuncConn name = "C5" direction = "output">
         <vnsRsMConnAtt tDn="uni/infra/mDev-Acme-ADC-1.0/mFunc-SLB/mConn-internal" />
     </vnsAbsFuncConn>

     <vnsAbsDevCfg>
         <vnsAbsFolder key="Network" name="Network" scopedBy="epg">
             <!-- Following scopes this folder to input terminal or Src Epg -->
           <vnsRsScopeToTerm tDn="uni/tn-tenant1/AbsGraph-G3/AbsTermNode-Output1/outtmnl"/>


             <!-- VIP address -->
             <vnsAbsFolder key="vip" name="vip" scopedBy="epg">
                 <vnsAbsParam name="vipaddress" key="vipaddress" value=""/>
             </vnsAbsFolder>

             <!-- SNIP address -->
             <vnsAbsFolder key="snip" name="snip" scopedBy="epg">
                 <vnsAbsParam name="snipaddress" key="snipaddress" value=""/>
             </vnsAbsFolder>
```

```
        </vnsAbsFolder>

        <vnsAbsFolder key="Service" name="Service" scopedBy="epg" cardinality="n">
          <vnsRsScopeToTerm tDn="uni/tn-tenant1/AbsGraph-G3/AbsTermNode-Output1/outtmnl"/>

            <vnsAbsParam name="servicename" key="servicename" value=""/>
            <vnsAbsParam name="servername" key="servername" value=""/>
            <vnsAbsParam name="serveripaddress" key="serveripaddress" value=""/>
        </vnsAbsFolder>
    </vnsAbsDevCfg>

    <vnsAbsFuncCfg>
        <vnsAbsFolder key="VServer" name="VServer" scopedBy="epg">
          <vnsRsScopeToTerm tDn="uni/tn-tenant1/AbsGraph-G3/AbsTermNode-Output1/outtmnl"/>

            <!-- Virtual Server Configuration -->
            <vnsAbsParam name="vip" key="vip" value=""/>
            <vnsAbsParam name="vservername" key="vservername" value=""/>
            <vnsAbsParam name="servicename" key="servicename"/>
            <vnsRsCfgToConn tDn="uni/tn-tenant1/AbsGraph-G3/AbsNode-Node2/AbsFConn-C4" />
        </vnsAbsFolder>
    </vnsAbsFuncCfg>
    <vnsRsNodeToMFunc tDn="uni/infra/mDev-Acme-ADC-1.0/mFunc-SLB"/>
</vnsAbsNode>
```

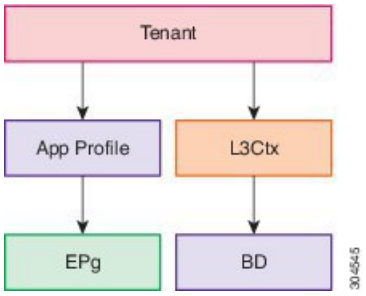
Configuration Parameters Inside Various Configuration MOs

An administrator can specify configuration parameters for a service function as part of various Application Policy Infrastructure Controller (APIC) MOs, such as EPG, tenant, BD, or AP. When a graph is instantiated, the APIC resolves the needed configuration for a graph by looking up the parameters from various places. At instantiation, parameter values are resolved and passed to the device script.

The flexibility of being able to keep configuration parameters inside various MOs allows an administrator to configure a single service graph and then use the graph for different tenants or end point groups (EPGs) with a different configuration for different tenants or EPGs.

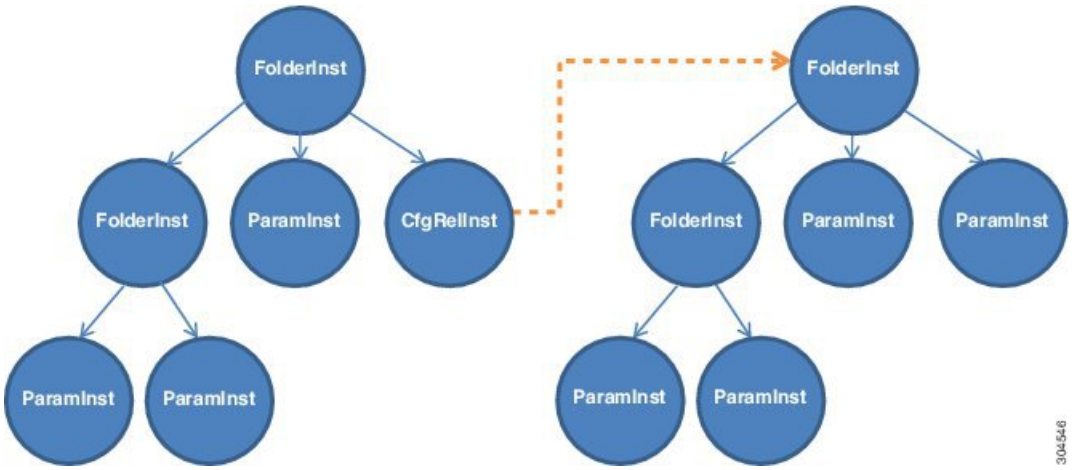
The following figure shows the hierarchy of an APIC MO.

Figure 36: Hierarchy of an APIC MO



The following figure shows configuration parameters inside various configuration MOs.

Figure 37: Configuration Parameters Inside Various Configuration MOs



FolderInst

FolderInst is a group of configuration items that can contain ParamInsts and other nested FolderInsts. A FolderInst has the following attributes:

Attribute	Description
Key	Defines the type of the configuration item. The key is defined in the device package and can never be overwritten. The key is used as a matching criterion as well as for validation.
ctrctNameOrLbl	Finds a matching FolderInst during parameter resolution. For a FolderInst to be used for parameter resolution, this property must match with the name of the contract that is associated with the service graph. Otherwise, this FolderInst is skipped and values are not used from this FolderInst. The value of this field can be "any" to allow this FolderInst to be used for all contracts.
graphNameOrLbl	Finds a matching FolderInst during parameter resolution. For a FolderInst to be used for parameter resolution, this property must match with the service graph name. Otherwise, this FolderInst is skipped and values are not used from this FolderInst. The value of this field can be “any” to allow this FolderInst to be used for all service graphs.
nodeNameOrLbl	Finds a matching FolderInst during parameter resolution. For a FolderInst to be used for parameter resolution, this property must match with the node name. Otherwise, this FolderInst is skipped and values are not used from this FolderInst. The value of this field can be “any” to allow this FolderInst to be used for all nodes in a service graph.

ParamInst

ParamInst is the basic unit of configuration parameters. ParamInst defines a single configuration parameter. As with a FolderInst, each ParamInst must have an equivalent MParam in the device specification. The APIC validates the specification to ensure that ParamInst has a corresponding MParam in the package. The value

of an ParamInst is validated using the validation method specified in the corresponding MParam. ParamInst has following attributes:

Attribute	Description
Key	Defines the type of the configuration item. The key is defined in the device package and can never be overwritten. The key is used as a matching criterion as well as for validation.
Value	Holds the value for a given configuration item. Values are not supported for MParam.

CfgRelInst

CfgRelInst has following attributes:

Attribute	Description
Key	Defines the type of the configuration item. The key is defined in the device package and can never be overwritten. The key is used as a matching criterion as well as for validation.
Value	Holds the path for the target FolderInst.

Example XML POST for an Application EPG With Configuration Parameters

The following XML example shows configuration parameters inside of the device package:

```
<fvAEPg dn="uni/tn-acme/ap-myApp/epg-app"  name="app">

    <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any" nodeNameOrLbl="any" key="Monitor"

       name="monitor1">
        <vnsRsFolderInstToMFolder tDn="uni/infra/mDev-Acme-ADC-1.0/mDevCfg/mFolder-Monitor"/>

          <vnsParamInst name="weight" key="weight" value="10"/>
     </vnsFolderInst>

    <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any" nodeNameOrLbl="any" key="Service"

       name="Service1">
         <vnsParamInst name="servicename" key="servicename" value="crpvgrtst02-8010"/>
         <vnsParamInst name="servicetype" key="servicetype" value="TCP"/>
         <vnsParamInst name="servername" key="servername" value="s192.168.100.100"/>
        <vnsParamInst name="serveripaddress" key="serveripaddress" value="192.168.100.100"/>

         <vnsParamInst name="serviceport" key="serviceport" value="8080"/>
         <vnsParamInst name="svrtimeout" key="svrtimeout" value="9000" />
         <vnsParamInst name="clttimeout" key="clttimeout" value="9000" />
         <vnsParamInst name="usip" key="usip" value="NO" />
         <vnsParamInst name="useproxyport" key="useproxyport" value="" />
         <vnsParamInst name="cip" key="cip" value="ENABLED" />
         <vnsParamInst name="cka" key="cka" value="NO" />
         <vnsParamInst name="sp" key="sp" value="OFF" />
         <vnsParamInst name="cmp" key="cmp" value="NO" />
         <vnsParamInst name="maxclient" key="maxclient" value="0" />
         <vnsParamInst name="maxreq" key="maxreq" value="0" />
         <vnsParamInst name="tcpb" key="tcpb" value="NO" />
         <vnsCfgRelInst name="MonitorConfig" key="MonitorConfig" targetName="monitor1"/>
     </vnsFolderInst>

    <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="G2" nodeNameOrLbl="any" key="Network"
```

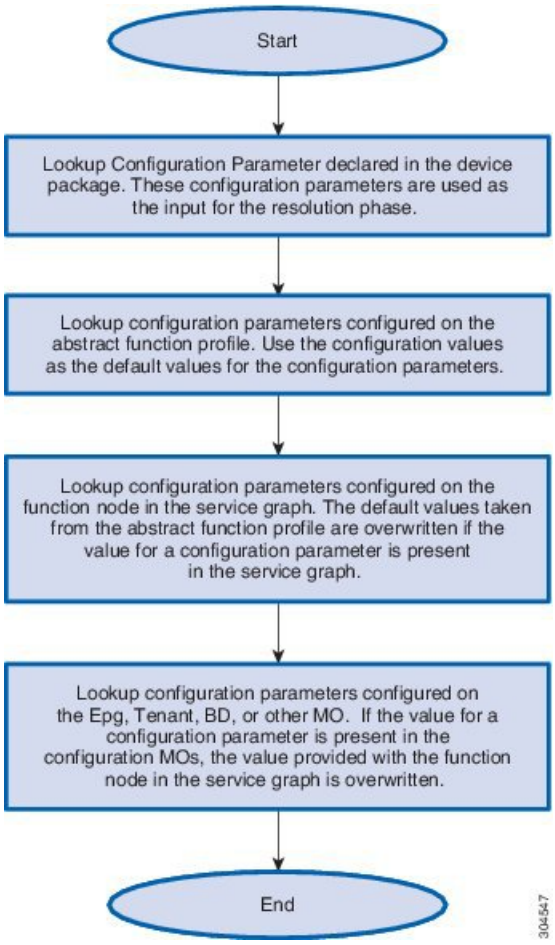
```
      name="Network">
       <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="G2" nodeNameOrLbl="any" key="vip"

          name="vip">
            <vnsParamInst name="vipaddress1" key="vipaddress" value="10.10.10.200"/>
        </vnsFolderInst>
        <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="G2" nodeNameOrLbl="any"
          devCtxLbl="C1" key="snip" name="snip1">
            <vnsParamInst name="snipaddress" key="snipaddress" value="192.168.1.200"/>
        </vnsFolderInst>
        <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="G2" nodeNameOrLbl="any"
          devCtxLbl="C2" key="snip" name="snip2">
    <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="G1" nodeNameOrLbl="any" key="Network"

      name="Network">
       <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="G1" nodeNameOrLbl="any" key="vip"

          name="vip">
            <vnsParamInst name="vipaddress1" key="vipaddress" value="10.10.10.100"/>
        </vnsFolderInst>
        <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="G1" nodeNameOrLbl="any"
          devCtxLbl="C1" key="snip" name="snip1">
            <vnsParamInst name="snipaddress" key="snipaddress" value="192.168.1.100"/>
        </vnsFolderInst>
        <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="G1" nodeNameOrLbl="any"
          devCtxLbl="C2" key="snip" name="snip2">
            <vnsParamInst name="snipaddress" key="snipaddress" value="192.168.1.101"/>
        </vnsFolderInst>
        <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="G1" nodeNameOrLbl="any"
          devCtxLbl="C3" key="snip" name="snip3">
            <vnsParamInst name="snipaddress" key="snipaddress" value="192.168.1.102"/>
        </vnsFolderInst>
     </vnsFolderInst>

     <!-- SLB Configuration -->
    <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any" nodeNameOrLbl="any" key="VServer"

      name="VServer">
        <!-- Virtual Server Configuration -->
        <vnsParamInst name="port" key="port" value="8010"/>
        <vnsParamInst name="vip" key="vip" value="10.10.10.100"/>
        <vnsParamInst name="vservername" key="vservername" value="crpvgrtst02-vip-8010"/>
        <vnsParamInst name="servicename" key="servicename" value="crpvgrtst02-8010"/>
        <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any" nodeNameOrLbl="any"
          key="VServerGlobalConfig" name="VServerGlobalConfig">
           <vnsCfgRelInst name="ServiceConfig" key="ServiceConfig" targetName="Service1"/>

            <vnsCfgRelInst name="VipConfig" key="VipConfig" targetName="Network/vip"/>
        </vnsFolderInst>
    </vnsFolderInst>
</fvAEPg>
```

Parameter Resolution

During graph instantiation, the Application Policy Infrastructure Controller (APIC) resolves the configuration parameters for each function in the service graph. After resolution completes, the parameter values are passed to the device script. The device script uses these parameter values to configure the service on the service appliance.

The following flow chart describes the parameter resolution steps.

Figure 38: Parameter Resolution



Looking Up an MO During Parameter Resolution

The Application Policy Infrastructure Controller (APIC) uses two main constructs while finding the suitable configuration MO to take the configuration parameters from.

RsScopeToTerm

The RsScopeToTerm relation for a function node or for an AbsFolder indicates the terminal node of the service graph that is connected with the configuration MOs that has parameters for the graph. The APIC uses the configuration MOs that are connected to the specified terminal node in RsScopeToTerm to find the graph configuration parameters.

If there is no RsScopeToTerm configuration specified, APIC uses the terminal connected to the provider EPG by default.

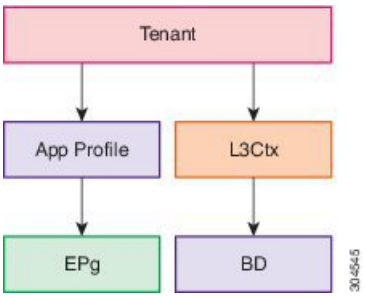
ScopedBy Attribute

The ScopedBy attribute is used to find the starting MO from which to resolve the parameter. For example, if scopedBy has a value of "Epg", the APIC starts the parameter resolution from the endpoint group. The APIC

then walks up in the hierarchy to resolve the parameters, walking to the application profile and then to the tenant to resolve the configuration parameters.

The following figure shows the hierarchy of an APIC MO.

Figure 39: Hierarchy of an APIC MO



About Role-Based Access Control Rule Enhancements

Layer 4 to Layer 7 policy configurations in a multi-tenant environment required administrator intervention to create certain objects that cannot be created by tenant administrators using the classic role-based access control (RBAC) domains and roles model definition. An Application Policy Infrastructure Controller (APIC) provides more granular RBAC privileges in the management information tree (MIT) such that you can grant tenant administrators the privileges that are required to create the objects. Tenant administrators can also create RBAC rules through self-service without administrator intervention to grant permissions for resources under their tenant subtree to other tenants and users in the system.

Role-Based Access Control Rule Architecture

The role-based access control (RBAC) rule has a boolean field `allowWrites` that enhances the role-based access control (RBAC) model to allow writeability rules. Without the `allowWrites` field, you can only define read RBAC rules.

The `RbacRule` class is defined as follows:

```
Class aaa:RbacRule (CONCRETE)
Encrypted: false
Exportable: true
Persistent: true
Configurable: true
Write Access: [aaa, admin]
Read Access: [aaa, admin]
```

An RBAC rule allows users from a security domain to read the subtree starting at a specific object.

```
DN FORMAT:   [1] uni/rbacdb/rule-{[objectDn]}-dom-{domain}
```

Table 2: aaa:RbacRule Properties Summary

Property	Type	Class	Description
aaa:Boolean	scalar:Enum8	allowWrites (aaa:RbacRule:allowWrites)	Read-write or read rule.

Property	Type	Class	Description
naming:Name	string:Basic	domain (aaa:RbacRule:domain)	The domain of the counts object. Overrides `aaa:ARbacRule:domain`.
reference:BinRef		objectDn (aaa:RbacRule:objectDn)	Overrides `aaa:ARbacRule:objectDn`.

The `PartialRbacRule` class is defined under the `fvTenant` class to allow tenants to create RBAC Rules (self-service). The `PartialRbacRule` class is defined as follows:

```
Class aaa:PartialRbacRule (CONCRETE)
Encrypted: false
Exportable: true
Persistent: true
Configurable: true
Write Access: [aaa, admin]
Read Access: [aaa, admin]
```

Table 3: aaa:PartialRbacRule Properties Summary

Property	Type	Class	Description
aaa:Boolean	scalar:Enum8	allowWrites (aaa:PartialRbacRule:allowWrites)	Read-write or read rule.
naming:Name	string:Basic	domain (aaa:PartialRbacRule:domain)	The domain of the counts object.
reference:BinRef		monPolDn (aaa:PartialRbacRule:monPolDn)	The monitoring policy attached to this observable object.
reference:BinRef		partialObjectDn (aaa:PartialRbacRule:partialObjectDn)	

The creation of the `PartialRbacRule` class by the tenant must be checked for a legal `partialObjectDn`. If the `partialObjectDn` lies under the tenant subtree, it is legal. Any distinguished names outside of the parent tenant subtree are not permitted.

The administrator can create `RbacRule`s that point to any distinguished name in the system. The tenant administrator can only create `PartialRbacRule`s that point to distinguished names that lie in that tenant administrator's tenant subtree.

Role-Based Access Control Rule System Flow

Either before, during, or after configuring the Layer 4 to Layer 7 policy, the tenant administrator can choose to create `PartialRbacRule`s that grant access to specific firewall and load balancer devices to their own tenant users. The access is implemented by creating `aaaDomain`s that represent each resource group that must be individually assigned. The following information provides an example setup:

Tenant	Acme

Users	acme-admin acme-firewall-1-admin acme-firewall-2-admin acme-loadbalancer-1-admin acme-loadbalancer-2-admin
Firewall Devices	Firewall1 Firewall2
Load Balancer Devices	LB1 LB2

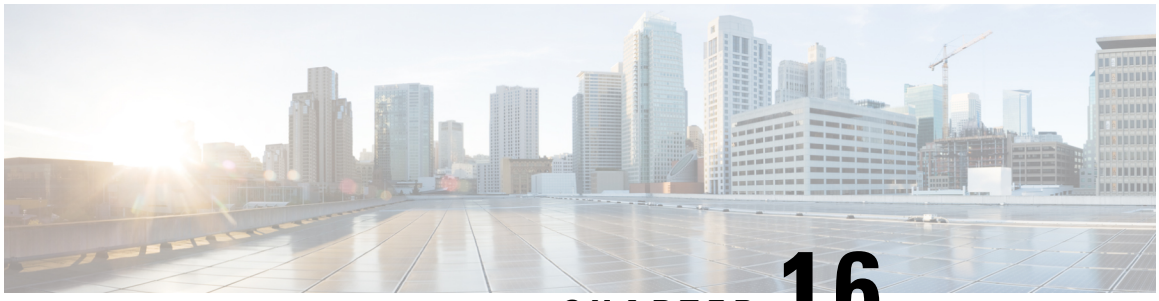
The tenant administrator user acme-admin wishes to create the devices Firewall1, Firewall2, LB1, and LB2. Full write permissions for each device need to be assigned on an individual user basis. For example, user acme-firewall-1-admin must only have write privileges to device Firewall1 policies, while user acme-loadbalancer-1-admin must only have write privileges to device LB1 policies. To accomplish this, the acme-admin user creates 4 `PartialRbacRule`s that grant the following access:

- Firewall1 distinguished name—writeable by domain acme-firewall1
- Firewall2 distinguished name—writeable by domain acme-firewall2
- LB1 distinguished name—writeable by domain acme-lb1
- LB2 distinguished name—writeable by domain acme-lb2

Users are then assigned the following privileges:

- User—acme-firewall-1-admin
  - Domain acme—read-all permissions
  - Domain acme-firewall1—tenant-admin/write
- User—acme-firewall-2-admin
  - Domain acme—read-all permissions
  - Domain acme-firewall2—tenant-admin/write
- User—acme-lb-1-admin
  - Domain acme—read-all permissions
  - Domain acme-lb1—tenant-admin/write
- User—acme-lb-2-admin
  - Domain acme—read-all permissions
  - Domain acme-lb2—tenant-admin/write

For any of the above 4 users, the domain acme's permissions allow them to read the acme tenant subtree, but not write any nodes. The domain acme-lb2's permissions of tenant-admin/write allow the user to write to the LB2 policy subtree only.



CHAPTER 16

Monitoring a Service Graph

- Monitoring a Service Graph Instance Using the GUI, on page 173
- Monitoring Service Graph Faults Using the GUI, on page 174
- Resolving Service Graph Faults, on page 174
- Monitoring a Virtual Device Using the GUI, on page 179
- Monitoring Device Cluster and Service Graph Status Using the NX-OS-Style CLI, on page 180

Monitoring a Service Graph Instance Using the GUI

After you configure a service graph template and attach the graph to an endpoint group (EPG) and a contract, you can monitor the service graph instance. Monitoring includes viewing the state of the graph instances, functions of a graph instance, resources allocated to a function, and parameters specified for a function.

Step 1 On the menu bar, choose **Tenants** > **All Tenants**.

Step 2 In the Work pane, double click the tenant's name for which you want to monitor its service graph.

Step 3 In the **Navigation** pane, choose **Tenant** *tenant_name* > **Services** > **L4-L7** > **Deployed Graph Instances**. The **Work** pane displays the following information about the active service graph instances:

Name	Description
Service Graph column	The name of the service graph template.
Contract column	The name of the contract that is shown in the service graph template.
Contained By column	The name of the network that contains the service graph template.
State column	The state of the service graph template. A state of **applied** means that the graph has been applied, and the graph policy is active within the fabric and the service device.
Description column	The description of the service graph.

Step 4 Expand the **Deployed Service Graphs** branch. The active service graph instances are listed under the branch.

Step 5 Click a service graph instance to view additional information about that instance in the **Work** pane. The default view is the topology of the graph. You can click one of the tabs in the **Work** pane to change the view for that graph.

Step 6 Expand the branch for one of the graph instances. The functions of the graph instance are listed under the instance.

Step 7 Click one of the functions to view additional information about that function in the **Work** pane. The default view is the policy of that function. You can click one of the tabs in the **Work** pane to change the view for that function. The **Work** pane displays the following information about the policy:

Name	Description
POLICY tab	The function's properties, resources allocated to the function, and the parameters of the function.
FAULTS tab	The issues that are happening on the function node.
HISTORY tab	The history of events that occurred on the function node.

Step 8 In the **Navigation** pane, click **Deployed Device**. The **Work** pane displays information about the device instances.

Monitoring Service Graph Faults Using the GUI

After you configure a service graph template and attach the graph to an endpoint group (EPG) and a contract, you can monitor a service graph template's faults.

Step 1 On the menu bar, choose **Tenants** > **All Tenants**.

Step 2 In the Work pane, double click the tenant's name for which you want to monitor its service graph.

Step 3 In the Navigation pane, choose **Tenant** *tenant_name* > **Services** > **L4-L7** > **Deployed Graph Instances**.

Step 4 Expand the branch for a graph instance for which you want to view its faults. The functions of the graph instance are listed under the instance.

Step 5 Click on one of the functions. By default, the **Work** pane shows the policy of that function.

Step 6 Click the **FAULTS** tab in the **Work** pane. The **Work** pane displays the faults of the function node.

Resolving Service Graph Faults

After you have observed one or more service graph template faults, resolving the issue depends on the fault. The following tables describe the faults and provide how to resolve faults.

Table 4: Connector Faults

Fault	CLI Label	Description and Resolution
missing-connection	connection associated with a connector not found	The configuration for a graph connector is invalid. The associated connection for the connector could not be found.

Fault	CLI Label	Description and Resolution
missing-nodeinst	NodeInst associated with a connector not found	The configuration for a graph connector is invalid. The associated NodeInst for the connector could not be found.
conn-nonrenderable	Graph connector could not be rendered.	The configuration for a graph connector is invalid. The graph could not be rendered.
invalid-bd	BD associated with a connector is not valid	The configuration for a graph connector is invalid. The associated bridge domain for the connector is not valid.
invalid-ctx	Ctx associated with a connector is not valid.	The configuration for a graph connector is invalid. The associated Ctx for the connector is not valid.
missing-peer-conn	Peer connector associated with a connector not found.	Configuration for a graph connector is invalid. The peer connector for the connection could not be found.

Table 5: AbsGraph and GraphInst Faults

Fault	CLI Label	Description and Resolution
invalid-abstract-graph-config	invalid abstract graph config	The abstract graph configuration is invalid.
missing-mandatory-param	mandatory param not found	A mandatory configuration parameter could not be resolved. Check the package for the mandatory parameter and make sure that AbsGraph has the parameter.
param-cardinality-error	invalid param cardinality	A configuration parameter does not meet cardinality requirements. Check if you specified multiple instances of a parameter without specifying cardinality=n.
epp-download-failure	epp download failure	Graph policies failed to download to the switch.
param-duplicate-name-failure	duplicate param name	Multiple identical copies of a parameter were found with the same name.
id-allocation-failure	id allocation failure	A unique network resource (either VLAN or VXLAN) could not be allocated.
missing-ldev	No cluster found	A cluster could not be found.

Fault	CLI Label	Description and Resolution
context-cardinality-violation-failure	invalid cluster context cardinality	The cluster does not support the required tenancy(multi-tenant or single tenant).
function-type-mismatch-failure	invalid function type	The function type is not supported for the selected device. Check if the AbsNode functype and resolved LDevVip function type match.
invalid-abstract-graph-config-param	invalid abstract graph config param	The abstract graph configuration parameter is invalid.
missing-mparam	No parameter definition found	A required parameter definition could not be found.
missing-abs-graph	no abs graph found	The abstract graph configuration is missing for the graph instance.
invalid-param-config	invalid param config	The parameter configuration is invalid.
invalid-param-scope	invalid parameter scope	The parameter scope is invalid. Check the vnsRsScopeToTerm parameter in the AbsGraph to see if parameter is correct.
invalid-ldev	Invalid cluster	The cluster configuration is invalid. Check the status of the resolved LDevVip and correct the fault.
missing-tenant	no tenant found	The tenant could not be found for the graph.
internal-error	internal error	An internal error occurred during graph processing.
resource-allocation-failure	resource allocation failure	A required resource could not be allocated during graph processing.
missing-abs-function	no abstract function found	The abstract function definition is missing.
param-validation-failed	param validation failure	The configuration parameter value is invalid.
missing-mconn	No connector found	A required connector could not be found.
cdev-missing-mgmt-ip	no mgmt ip found for cdev	The management IP address could not be found for concrete device. Check if vnsCMgmt is present for the resolved vnsCDev.
invalid-graphinst	invalid graphinst config	The graph instance is invalid.
missing-interface	no interface found	An interface could not be found.

Fault	CLI Label	Description and Resolution
missing-bd	no bd found	A bridge domain could not be found.
missing-terminal	Terminal node is missing a terminal	Terminal node is missing a terminal. Check the terminal node settings.
missing-namespace	no vlan/vxlan namespace found	The namespace that is needed to allocate the VLAN or VXLAN is missing. Verify that the resolved vnsLDevVip has the phyDomp parameter or the vmmDomp parameter configured that has a relation to the resolved fvnsVlanInstp.
missing-mfunc	No function found in device package	A required function could not be found in the device package. Verify that all AbsNode function types are present in the package.
missing-lif	no cluster interface found	A required cluster interface could not be found. Verify that the vnsLIf parameter in vnsLDevVip is configured correctly.
invalid-absfunc-profile	Abstract Function Profile config is invalid	The abstract function profile configuration is invalid. This fault may be due to an invalid configuration parameter that is specified in the profile.
missing-cdev	No device found	The concrete device could not be found in the cluster. Verify that a valid vnsCDev is present under the resolved vnsLDevVip.
inappropriate-devfolder	Illegal folder in configuration	No corresponding folder was found in the device package.
invalid-devctx	Device context is not legal for this folder	The device package does not allow specifying a device context for this folder.
insufficient-devctx	Folder must have one value for each associated CDev	The folder is concrete device specific. The folder must have at least one value for each concrete device.
cdev-missing-cif	No interface defined	A concrete device must have at least one interface defined.

Fault	CLI Label	Description and Resolution
cdev-missing-pathinfo	Missing path for interface	For a physical service appliance, we must know to which leaf ports the interface is connected. Verify that the vnsCifPathAtt parameter is present for all vnsCIf under the resolved vnsCDev.
missing-cif	Device interfaces does not match cluster	The device interfaces should match the interfaces configured for their cluster. Verify that the vnsCIf parameter and the vnsLIf parameter are present under the resolved vnsLDevVip.
ldevvip-missing-mgmt-ip	No Mgmt ip found for LDevVip	The management IP address could not be found for LDevVip.
lif-invalid-MIf	LIf has an invalid MifLbl	The MIfLbl contained by LIf is not present in the device package.
lif-invalid-CIf	LIf has an invalid CIf	The CIf contained by LIf is not present. Check the concrete device and CIf settings.
missing-function-node	Abstract graph missing function node	An abstract graph must have at least one function node.
graph-loop-detected	Abstract graph config has a loop	The abstract graph configuration is invalid. The configuration has a loop.
gothrough-routing-enabled-both	Both the legs of go through node has routing enabled	Both the legs of the go through node have routing enabled.
invalid-terminal-nodes	Abstract graph has invalid number of terminal nodes	An abstract graph must have at least two terminal nodes.
missing-ldev-ctx	No device context found for LDev	The device context for the device could not be found. Verify that vnsLDevCtx has values that match the contract, graph and node.
arp-flood-enabled	ARP flood is enabled on the management end point group	ARP flood must be disabled for the management endpoint group.
folderinst-validation-failed	FolderInst has key, that is not found in MFolder	The FolderInst's key and value should honor MFolder specifications.
paraminst-validation-failed	ParamInst has key and/or value, that are not found in MParam	ParamInst's key and value should honor MParam specifications.
invalid-mfolder	FolderInst points to an invalid MFolder	FolderInst must point to a valid MFolder.

Fault	CLI Label	Description and Resolution
invalid-mparam	ParamInst points to an invalid MParam	ParamInst must point to a valid MParam.
devfolder-validation-failed	DevFolder has key, that is not found in MFolder	DevFolders key and value should honor MFolder specifications.
devparam-validation-failed	DevParam has key and/or value, that are not found in MParam	DevParam's key and value should honor MParam specifications
cdev-missing-virtual-info	Virtual Object Info is missing in CDev	Virtual object information must be provided if LDevVip is of type Virtual.
invalid-rsmconnatt	Relationship to metaconnector is invalid	Correct the metaconnector DN and ensure it binds to the correct MDev hierarchy.

Monitoring a Virtual Device Using the GUI

After you configure a service graph template and attach the graph to an endpoint group (EPG) and a contract, you can monitor the virtual devices of a tenant. Monitoring the virtual devices tells you what devices are in use, which VLANs are configured for a device, the parameters passed to the devices, the statistics of the devices, and the health of the devices.

Step 1 On the menu bar, choose **Tenants** > **All Tenants**.

Step 2 In the Work pane, double click the tenant's name for which you want to monitor its service graph.

Step 3 In the Navigation pane, choose **Tenant** ***tenant_name*** > **Services** > **L4-L7** > **Deployed Devices**.

Step 4 Click on one of the deployed devices. By default, the **Work** pane shows the policy of that deployed device. You can click the tabs in the **Work** pane to change the view. The tabs display the following information about the virtual device:

Tab	Description
POLICY tab	The device that is in use, the VLANs that are configured within the device, and the parameters that have been passed to the devices.
OPERATIONAL tab	The statistics that are being received from the various devices.
HEALTH tab	The health of the devices.

Monitoring Device Cluster and Service Graph Status Using the NX-OS-Style CLI

The commands in this section provide examples of how to monitor device cluster and service graph status using the NX-OS-style CLI.

Showing the Operation Information of a Device Cluster

The following command shows the operational information of a device cluster:

```
show l4l7-cluster tenant tenant_name cluster device_cluster_name
```

Example:

```
apic1# show l4l7-cluster tenant HA_Tenant1 cluster Firewall
tenant-graph : HA_Tenant1-g2,HA_Tenant1-g1


Device Cluster    : Firewall
Cluster Interface : consumer1
Encap             : vlan-501
Pctag             : 32773
Devices           : FW2(int),FW1(int)
Graphs            : HA_Tenant1-g1
Contracts         : HA_Tenant1-c1


Device Cluster    : Firewall
Cluster Interface : provider1
Encap             : vlan-502
Pctag             : 32774
Devices           : FW2(ext),FW1(ext)
Graphs            : HA_Tenant1-g1
Contracts         : HA_Tenant1-c1
```

Showing the Operation Status of a Device Cluster

The following command shows the operation status of a device cluster:

```
apic1# show l4l7-graph tenant tenant_name [graph graph_name]
```

Examples:

The following example gives high-level output of the status of the `HA_Tenant1` tenant:

```
apic1# show l4l7-graph tenant HA_Tenant1
Graph           : g1
Total Instances : 1
Encaps Used     : vlan-501,vlan-502,vlan-503,vlan-504
Device Used     : uni/tn-HA_Tenant1/lDevVip-Firewall

Graph           : g2
Total Instances : 1
Encaps Used     : vlan-501,vlan-502,vlan-503,vlan-504
Device Used     : uni/tn-HA_Tenant1/lDevVip-Firewall
```

The following example gives detailed output of the status of the `g1` service graph that is associated with the `HA_Tenant1` tenant:

```
apic1# show l4l7-graph tenant HA_Tenant1 graph g1
Graph           : HA_Tenant1-g1
Graph Instances : 1

Consumer EPg  : HA_Tenant1-consEPG1
Provider EPg  : HA_Tenant1-provEPG1
Contract Name : HA_Tenant1-c1
Config status : applied

Function Node Name : Node1
 Connector   Encap       Bridge-Domain  Device Interface
 ----------  ----------  ----------     --------------------
 consumer    vlan-3001   provBD1        consumer
 provider    vlan-3335   consBD1        provider
```

Showing the Faults of a Device Cluster

The following command shows the faults of a device cluster:

```
show faults l4l7-cluster
```

Example:

```
apic1# show faults l4l7-cluster
Code            : F0772
Severity        : minor
Last Transition : 2015-09-01T01:41:13.767+00:00
Lifecycle       : soaking-clearing
Affected object : uni/tn-ts1/lDevVip-d1/lIf-ext/fault-F0772
Description     : LIf configuration ext for L4-L7 Devices d1 for tenant ts1
                  is invalid.

Code            : F1085
Severity        : cleared
Last Transition : 2015-09-01T01:39:04.696+00:00
Lifecycle       : retaining
Affected object : uni/tn-ts1/lDevVip-d1/rsmDevAtt/fault-F1085
Description     : Failed to form relation to MO uni/infra/mDev-CiscoInternal-
                  NetworkOnly-1.0 of class vnsMDev

Code            : F1690
Severity        : minor
Last Transition : 2015-09-01T01:39:04.676+00:00
Lifecycle       : soaking
Affected object : uni/tn-ts1/lDevVip-d1/vnsConfIssue-missing-
                  namespace/fault-F1690
Description     : Configuration is invalid due to no vlan/vxlan namespace
                  found
```

Showing the Faults of a Service Graph

The following command shows the faults of a service graph:

```
show faults l4l7-graph
```

Example:

```
apic1# show faults l4l7-graph
Code            : F1690
Severity        : minor
Last Transition : 2015-11-25T20:07:33.635+00:00
Lifecycle       : raised
DN              : uni/tn-HA_Tenant1/AbsGraph-WebGraph/vnsConfIssue-invalid-
                  abstract-graph-config-param/fault-F1690
```

```
Description     : Configuration is invalid due to invalid abstract graph
                  config param
```

Showing the Running Configuration of a Device Cluster

The following command shows the running configuration of a device cluster:

```
show running-config tenant tenant_name l4l7 cluster
```

Example:

```
apic1# show running-config tenant common l4l7 cluster
# Command: show running-config tenant common l4l7 cluster
# Time: Thu Nov 26 00:35:59 2015
  tenant common
    l4l7 cluster name ifav108-asa type physical vlan-domain phyDom5 service FW function
go-through
      cluster-device C1
      cluster-interface consumer_1
        member device C1 device-interface port-channel1
          interface vpc VPCPolASA leaf 103 104
          exit
        exit
      cluster-interface provider_1
        member device C1 device-interface port-channel1
          interface vpc VPCPolASA leaf 103 104
          exit
        exit
      exit
```

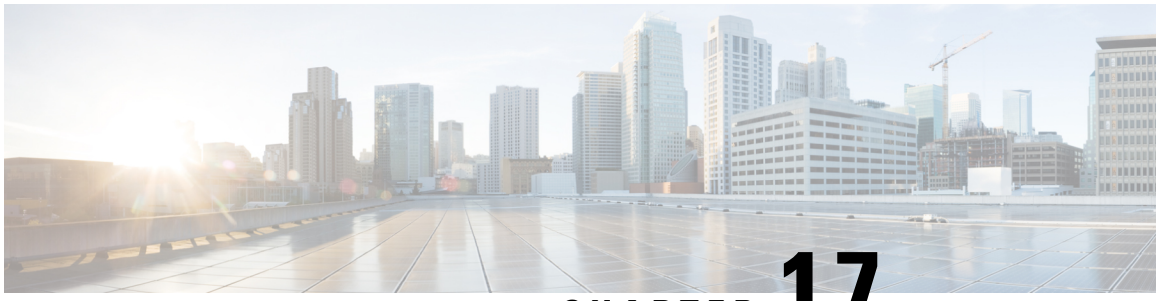
Showing the Running Configuration of a Service Graph

The following command shows the running configuration of a service graph:

```
show running-config tenant tenant_name l4l7 graph
```

Example:

```
apic1# show running-config tenant common l4l7 graph
# Command: show running-config tenant common l4l7 graph
# Time: Thu Nov 26 00:35:59 2015
  tenant T1
    l4l7 graph Graph-Citrix contract Contract-Citrix
     service N1 device-cluster-tenant common device-cluster ifav108-citrix mode ADC_ONE_ARM

        connector provider cluster-interface pro
          bridge-domain tenant common name BD4-Common
          exit
        connector consumer cluster-interface pro
          bridge-domain tenant common name BD4-Common
          exit
        exit
      connection C1 terminal consumer service N1 connector consumer
      connection C2 terminal provider service N1 connector provider
      exit
```



CHAPTER 17

Configuring Multi-Tier Application with Service Graph

- About Multi-Tier Application with Service Graph, on page 183
- Creating a Multi-Tier Application Profile Using the GUI, on page 183

About Multi-Tier Application with Service Graph

The Multi-Tier Application with Service Graph Quick Start dialog provides a consolidated method of configuring service graph components such as bridge domains, EPGs, VRFs, services, and contracts. As opposed to configuring each object in different locations in the Cisco APIC, the Quick Start dialog gathers the necessary configurations and combines them into a simple, organized step-by-step process.

Creating a Multi-Tier Application Profile Using the GUI

Before you begin

Configure the following objects before or, if available, while performing the procedure:

- **Tenants:** Configure at least one tenant before performing the procedure.
- **VMM Domain Profile:** If you will use virtual service devices, configure a Virtual Machine Manager (VMM) domain profile and a VM in the Layer 4 to Layer 7 device cluster (on which the device is hosted).
- **External Routed Network:** If you will connect a service device to an external routed network, configure a Layer 3 outside (L3Out) network.

Step 1 Access the Quick Start **Multi-Tier Application** dialog:

a) On the menu bar, click **Tenant** > **All Tenants**.
b) In the All Tenants Work pane, double-click the tenant's name.
c) In the Navigation pane, choose **Tenant** *tenant_name* > **Quick Start** > **Multi-tier Application**.
d) In the work pane, click **Configure Multi-tier Application**.
The **Create Application Profile** dialog appears.
e) Click **Start**.

Step 2 In the **STEP 2 > EPGs** dialog, configure the basics of the profile and design your Bridge Domain and EPGs:

a) In the **Application Profile** field, enter a unique name for the profile.

b) (Optional) If one or more devices in this profile are to be virtual, choose a Virtual Machine Manager (VMM) domain profile from the **VMM Domain Profile** drop-down list.

Note A VMM domain profile must be created (**Virtual Networking** > **VMM Domains**) prior to attempting this step in order for it to appear and be selected in the **VMM Domain Profile** drop-down list.

c) (Optional) If the consumer or provider EPG belongs to an external routed network, choose the network from the drop-down list for the **Consumer L3 Outside** and/or the **Provider L3 Outside** field(s).

Note An external routed network must be created (**Tenants** > *tenant* > **Networking** > **External Routed Networks**) prior to attempting this step in order for it to appear and be selected in the **L3 Outside** drop-down lists.

d) For the Bridge Domain buttons, determine if the EPG gateway IP address will be a single shared subnet or will be configured per EPG.

If you chose **Shared**, the **Shared Gateway IP** field appears. If you chose **Per EPG**, continue with step f.

e) If you chose **Shared** from the **Bridge Domain** buttons, enter the IPv4 address of the gateway to be shared by the EPGs in the **Shared Gateway IP** field.

f) In the Application Tiers (EPGs) **Name** field, enter a name for the EPG.

g) If you chose **Per EPG** from the **Bridge Domain** buttons, enter the IPv4 address of the gateway to be used by the EPG. If you chose **Shared** from the **Bridge Domain** buttons, the IP address that you entered in the **Shared Gateway IP** field is displayed.

h) (Optional) Click **+** to add another EPG and configure the EPG according to step g. Repeat this step if a third EPG is required.

i) Click **Next**.

Step 3 In the **STEP 3 > Services** dialog, optionally configure the inclusion of services adjacent to your EPGs:

a) (Optional) Put a check in the **Share same device** box to share the firewall or load balancer devices across all EPGs.

b) (Optional) Between each EPG, select the firewall (**FW**) or load balancer (**ADC**) device to include in this profile.

c) (Optional) If you add more than one device between an EPG, click **< Toggle >** to reposition the devices.

d) Click **Next**.

Step 4 (Firewall and Load Balancer) In the **STEP 4 >** dialog and the Firewall or Load Balancer Configuration section, configure service devices:

a) For the **Device Type** buttons, choose **Physical** or **Virtual**.

b) If you chose **Physical** for the **Device Type**, choose a domain from the **Physical Domain** drop-down list. If you chose **Virtul** for the **Device Type**, choose a domain from the **VMM Domain** drop-down list and the virtual machine (VM) on which the device is hosted from the **Device 1 VM** drop-down list.

c) For the **Node Type** buttons, chhoose **One-Arm** or **Two-Arm**. This determines if the device has only a consumer connector (one-arm) or consumer and provider connectors (two-arm).

d) For the **View** buttons, choose **Single Node** or **HA Node**. If you chose **HA Node**, a second interface (physical devices) or a second VNIC (virtual devices) is included in the connector configuration. for virtual devices, you must also choose a second virtual machine.

Step 5 (Firewall only) In the **STEP 4 >** dialog and the Consumer and Provider section, configure the firewall consumer and provider connectors:

a) In the **IP** field, for a physical device, enter the consumer/provider interface IP address of the Layer 4 to Layer 7 policy based redirect policy for firewall devices. For a virtual device, enter the consumer/provider interface IP address.

b) In the **MAC** field, enter MAC address of the LLayer 4 to Layer 7 policy based redirect policy for firewall devices.

c) In the **Gateway IP** field, enter the route gateway IP address.
d) For a physical device, in the **Device 1 Interface** drop-down list, choose an interface. For a virtual device, in the **Device 1 VNIC** drop-down list, choose a vNIC. If you chose **HA Node** for from the **View** buttons, you must choose a second vNIC in the **Device 2 VNIC** drop-down list.
e) (Physical device only) In the **Encap** field, enter the port encapsulation for the interface.

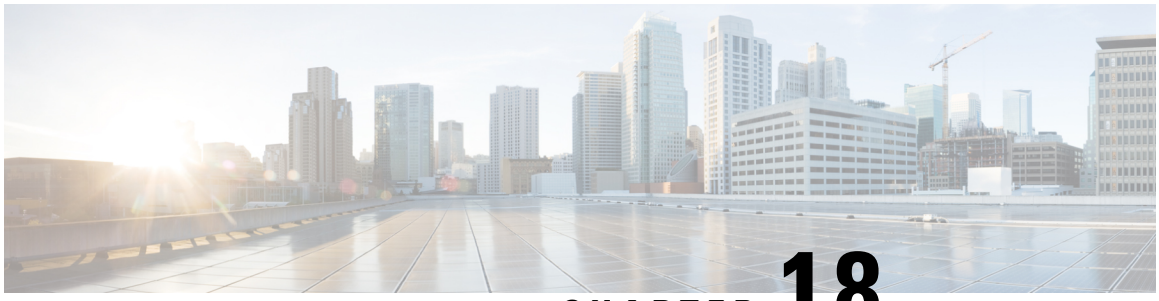
Step 6 (Load Balancer only) In the **STEP 4 >** dialog and the Consumer and Provider section, configure load balancer consumer and provider connectors:

a) In the **Gateway IP** field, enter the route gateway IP address.
b) For a physical device, in the **Device 1 Interface** drop-down list, choose an interface. For a virtual device, in the **Device 1 VNIC** drop-down list, choose a vNIC. If you chose **HA Node** for from the **View** buttons, you must choose a second vNIC in the **Device 2 VNIC** drop-down list.
c) (Physical device only) In the **Encap** field, enter the port encapsulation for the interface.
d) Leave the check in the **L3 Destination (VIP)** box to terminate L3 traffic on the connector. Remove the check if the connector is not an L3 destination.

Note The default for this parameter is enabled (checked). However, this setting is not considered if policy-based redirect is configured on the interface.

Step 7 If you have any additional devices to configure, click **Next** and repeat steps 4 through 6 for each device.

Step 8 Click **Finish**.



CHAPTER 18

Configuring Administrator Roles for Managing a Service Configuration

- About Privileges, on page 187
- Configuring a Role for Device Management, on page 188
- Configuring a Role for Service Graph Template Management, on page 188
- Configuring a Role for Uploading Device Package, on page 188
- Configuring a Role for Exporting Devices, on page 188

About Privileges

You can grant privileges to the roles that you configure in the Application Policy Infrastructure Controller (APIC). Privileges determine what tasks a role is allowed to perform. You can grant the following privileges to the administrator roles:

Privilege	Description
nw-svc-policy	The network service policy privilege enables you to do the following: • Create a service graph template • Attach a service graph template to an application endpoint group (EPG) and a contract • Monitor a service graph
nw-svc-device	The network service device privilege enables you to do the following: • Create a device • Create a concrete device • Create a device context

Note Only the infrastructure administrator can upload a device package to the Cisco APIC.

Configuring a Role for Device Management

To enable a role to manage devices, you must grant the following privilege to that role:

- nw-svc-device

Configuring a Role for Service Graph Template Management

To enable a role to manage service graph templatess, you must grant the following privilege to that role:

- nw-svc-policy

Configuring a Role for Uploading Device Package

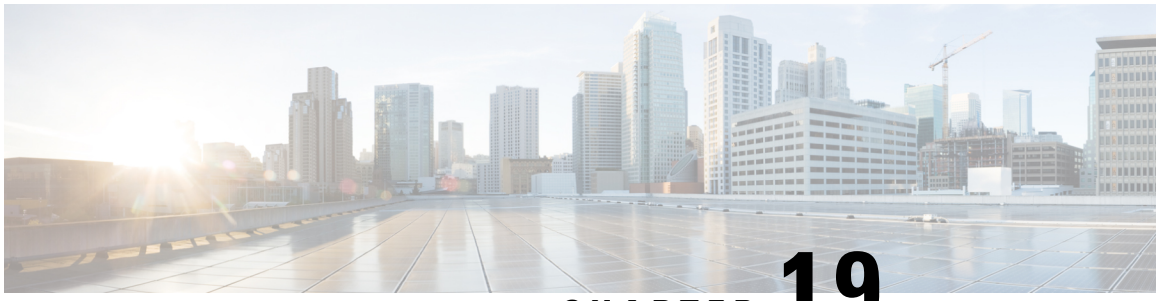
A device package can be uploaded only with the APIC infra admin privilege. Infra admin uploads the device packages. All other tenant administrators have read-only access to the device package. Tenant administrators can access and use various functions available from the device package.

Configuring a Role for Exporting Devices

Devices can be exported to enable sharing of devices among tenants. A tenant with the role **nw-device** can create a device. If the tenant that owns the device wants to share these with another tenant, the sharing requires the **nw-svc-devshare** privilege.

The **nw-svc-devshare** privilege allows a tenant to be able to export devices.

Note To be able to use imported devices, other tenants that have imported devices need to have the **nw-svc-policy** privilege.



CHAPTER 19

Developing Automation

- About the REST APIs, on page 189
- Examples of Automating Using the REST APIs, on page 189

About the REST APIs

Automation relies on the Application Policy Infrastructure Controller (APIC) northbound Representational State Transfer (REST) APIs. Anything that can be done through the Cisco APIC GUI can also be done using XML-based REST POSTs using the northbound APIs. For example, you can monitor events through those APIs, dynamically enable EPGs, and add policies.

You can also use the northbound REST APIs to monitor for notifications that a device has been brought onboard, and to monitor faults. In both cases, you can monitor events that trigger specific actions. For example, if you see faults that occur on a specific application tier and determine that there is a loss of connectivity and a leaf node is going down, you can trigger an action to redeploy those applications somewhere else. If you have certain contracts on which you detect packet drops occurring, you could enable some copies of those contracts on the particular application. You can also use a statistics monitoring policy, where you monitor certain counters because of issues that have been reported.

For information on how to construct the XML files submitted to the Cisco APIC northbound API, see *Cisco APIC Layer 4 to Layer 7 Device Package Development Guide*.

The following Python APIs, defined in the *Cisco APIC Management Information Model Reference* can be used to submit REST POST calls using the northbound API:

- `vns:LDevVip`: Upload a device cluster
- `vns:CDev`: Upload a device
- `vns:LIf`: Create logical interfaces
- `vns:AbsGraph`: Create a graph
- `vz:BrCP`: Attach a graph to a contract

Examples of Automating Using the REST APIs

This section contains examples of using the REST APIs to automate tasks.

The following REST request creates a tenant with a broadcast domain, a Layer 3 network, application endpoint groups, and an application profile:

```
<polUni>
    <fvTenant dn="uni/tn-acme" name="acme">

        <!—L3 Network-->
        <fvCtx name="MyNetwork"/>

        <!-- Bridge Domain for MySrvr EPG -->
        <fvBD name="MySrvrBD">
            <fvRsCtx tnFvCtxName="MyNetwork"/>
            <fvSubnet ip="10.10.10.10/24">
            </fvSubnet>
        </fvBD>

        <!-- Bridge Domain for MyClnt EPG -->
        <fvBD name="MyClntBD">
            <fvRsCtx tnFvCtxName="MyNetwork"/>
            <fvSubnet ip="20.20.20.20/24">
            </fvSubnet>
        </fvBD>

        <fvAp dn="uni/tn-acme/ap-MyAP" name="MyAP">

            <fvAEPg dn="uni/tn-acme/ap-MyAP/epg-MyClnt" name="MyClnt">
                <fvRsBd tnFvBDName="MySrvrBD"/>
                <fvRsDomAtt tDn="uni/vmmp-Vendor1/dom-MyVMs"/>
                <fvRsProv tnVzBrCPName="webCtrct"> </fvRsProv>
                <fvRsPathAtt tDn="topology/pod-1/paths-17/pathep-[eth1/21]"
                  encap="vlan-202"/>
                <fvRsPathAtt tDn="topology/pod-1/paths-18/pathep-[eth1/21]"
                  encap="vlan-202"/>
           </fvAEPg>

            <fvAEPg dn="uni/tn-acme/ap-MyAP/epg-MySRVR" name="MySRVR">
                <fvRsBd tnFvBDName="MyClntBD"/>
                <fvRsDomAtt tDn="uni/vmmp-Vendor1/dom-MyVMs"/>
                <fvRsCons tnVzBrCPName="webCtrct"> </fvRsCons>
                <fvRsPathAtt tDn="topology/pod-1/paths-17/pathep-[eth1/21]"
                  encap="vlan-203"/>
                <fvRsPathAtt tDn="topology/pod-1/paths-18/pathep-[eth1/21]"
                  encap="vlan-203"/>
            </fvAEPg>
        </fvAp>
    </fvTenant>
</polUni>
```

The following REST request creates a VLAN namespace:

```
<polUni>
    <infraInfra>
        <fvnsVlanInstP name="MyNS" allocMode="dynamic">
            <fvnsEncapBlk name="encap" from="vlan-201" to="vlan-300"/>
        </fvnsVlanInstP>
    </infraInfra>
</polUni>
```

The following REST request creates a VMM domain:

```
<polUni>
    <vmmProvP vendor="Vendor1">
        <vmmDomP name="MyVMs">
            <infraRsVlanNs tDn="uni/infra/vlanns-MyNS-dynamic"/>
            <vmmUsrAccP name="admin" usr="administrator" pwd="in$1eme"/>
```

```
            <vmmCtrlrP name="vcenter1" hostOrIp="192.168.64.186">
                <vmmRsAcc tDn="uni/vmmp-Vendor1/dom-MyVMs/usracc-admin"/>
            </vmmCtrlrP>
        </vmmDomP>
    </vmmProvP>
</polUni>
```

The following REST request creates a physical domain:

```
<polUni>
    <physDomP name="phys">
        <infraRsVlanNs tDn="uni/infra/vlanns-MyNS-dynamic"/>
    </physDomP>
</polUni>
```

The following REST request creates a managed device cluster:

```
<polUni>
    <fvTenant dn="uni/tn-acme" name="acme">
        <vnsLDevVip name="ADCCluster1" contextAware=1>
            <vnsRsMDevAtt tDn="uni/infra/mDev-Acme-ADC-1.0"/>
            <vnsRsDevEpg tDn="uni/tn-acme/ap-services/epg-ifc"/>
            <vnsRsALDevToPhysDomP tDn="uni/phys-phys"/>
            <vnsCMgmt name="devMgmt" host="42.42.42.100" port="80"/>
            <vnsCCred name="username"value="admin"/>
            <vnsCCredSecret name="password" value="admin"/>
        </vnsLDevVip>
    </fvTenant>
</polUni>
```

The following REST request creates an unmanaged device cluster:

```
<polUni>
    <fvTenant name="HA_Tenant1">
        <vnsLDevVip name="ADCCluster1" devtype="VIRTUAL" managed="no">
            <vnsRsALDevToDomP tDn="uni/vmmp-VMware/dom-mininet"/>
        </vnsLDevVip>
    </fvTenant>
</polUni>
```

The following REST request creates a device cluster context:

```
<polUni>
    <fvTenant dn="uni/tn-acme" name="acme">
        <vnsLDevCtx ctrctNameOrLbl="webCtrct" graphNameOrLbl="G1" nodeNameOrLbl="Node1">
            <vnsRsLDevCtxToLDev tDn="uni/tn-acme/lDevVip-ADCCluster1"/>
            <vnsLIfCtx connNameOrLbl="ssl-inside">
                <vnsRsLIfCtxToLIf tDn="uni/tn-acme/lDevVip-ADCCluster1/lIf-int"/>
            </vnsLIfCtx>
            <vnsLIfCtx connNameOrLbl="any">
                <vnsRsLIfCtxToLIf tDn="uni/tn-acme/lDevVip-ADCCluster1/lIf-ext"/>
            </vnsLIfCtx>
        </vnsLDevCtx>
    </fvTenant>
</polUni>
```

The following REST request creates a device cluster context used in route peering:

```
<polUni>
    <fvTenant dn="uni/tn-coke{{tenantId}}" name="coke{{tenantId}}">
        <vnsRtrCfg name="Dev1Ctx1" rtrId="180.0.0.12"/>
            <vnsLDevCtx ctrctNameOrLbl="webCtrct1" graphNameOrLbl="WebGraph"
              nodeNameOrLbl="FW">
                <vnsRsLDevCtxToLDev tDn="uni/tn-tenant1/lDevVip-Firewall"/>
                <vnsRsLDevCtxToRtrCfg tnVnsRtrCfgName="FwRtrCfg"/>
                <vnsLIfCtx connNameOrLbl="internal">
                  <vnsRsLIfCtxToInstP tDn="uni/tn-tenant1/out-OspfInternal/instP-IntInstP"
```

```
                    status="created,modified"/>
                <vnsRsLIfCtxToLIf tDn="uni/tn-tenant1/lDevVip-Firewall/lIf-internal"/>
            </vnsLIfCtx>
            <vnsLIfCtx connNameOrLbl="external">
               <vnsRsLIfCtxToInstP tDn="uni/tn-common/out-OspfExternal/instP-ExtInstP"

                    status="created,modified"/>
                <vnsRsLIfCtxToLIf tDn="uni/tn-tenant1/lDevVip-Firewall/lIf-external"/>
        </vnsLIfCtx>
    </vnsLDevCtx>
</fvTenant>
</polUni>
```

Note For information about configuring external connectivity for tenants (a Layer 3 outside), see the *Cisco APIC Basic Configuration Guide*.

The following REST request adds a logical interface in a device cluster:

```
<polUni>
    <fvTenant dn="uni/tn-acme" name="acme">
        <vnsLDevVip name="ADCCluster1">
            <vnsLIf name="C5">
                <vnsRsMetaIf tDn="uni/infra/mDev-Acme-ADC-1.0/mIfLbl-outside"/>
                <vnsRsCIfAtt tDn="uni/tn-acme/lDevVip-ADCCluster1/cDev-ADC1/cIf-int"/>
            </vnsLIf>
            <vnsLIf name="C4">
                <vnsRsMetaIf tDn="uni/infra/mDev-Acme-ADC-1.0/mIfLbl-inside"/>
                <vnsRsCIfAtt tDn="uni/tn-acme/lDevVip-ADCCluster1/cDev-ADC1/cIf-ext"/>
            </vnsLIf>
        </vnsLDevVip>
    </fvTenant>
</polUni>
```

The following REST request adds a concrete device in a physical device cluster:

```
<polUni>
    <fvTenant dn="uni/tn-acme" name="acme">
        <vnsLDevVip name="ADCCluster1">
            <vnsCDev name="ADC1" devCtxLbl="C1">
                <vnsCIf name="int">
                    <vnsRsCIfPathAtt tDn="topology/pod-1/paths-17/pathep-[eth1/22]"/>
                </vnsCIf>
                <vnsCIf name="ext">
                    <vnsRsCIfPathAtt tDn="topology/pod-1/paths-17/pathep-[eth1/21]"/>
                </vnsCIf>
                <vnsCIf name="mgmt">
                    <vnsRsCIfPathAtt tDn="topology/pod-1/paths-17/pathep-[eth1/20]"/>
                </vnsCIf>
                <vnsCMgmt name="devMgmt" host="172.30.30.100" port="80"/>
                <vnsCCred name="username" value="admin"/>
                <vnsCCredSecret name="password" value="admin"/>
            </vnsCDev>
            <vnsCDev name="ADC2" devCtxLbl="C2">
                <vnsCIf name="int">
                    <vnsRsCIfPathAtt tDn="topology/pod-1/paths-17/pathep-[eth1/23]"/>
                </vnsCIf>
                <vnsCIf name="ext">
                    <vnsRsCIfPathAtt tDn="topology/pod-1/paths-17/pathep-[eth1/24]"/>
                </vnsCIf>
                <vnsCIf name="mgmt">
                    <vnsRsCIfPathAtt tDn="topology/pod-1/paths-17/pathep-[eth1/30]"/>
```

```
                </vnsCIf>
                <vnsCMgmt name="devMgmt" host="172.30.30.200" port="80"/>
                <vnsCCred name="username" value="admin"/>
                <vnsCCredSecret name="password" value="admin"/>
            </vnsCDev>
        </vnsLDevVip>
    </fvTenant>
</polUni>
```

The following REST request adds a concrete device in a virtual device cluster:

```
<polUni>
    <fvTenant dn="uni/tn-coke5" name="coke5">
        <vnsLDevVip name="Firewall5" devtype="VIRTUAL">
            <vnsCDev name="ASA5" vcenterName="vcenter1" vmName="ifav16-ASAv-scale-05">
                <vnsCIf name="Gig0/0" vnicName="Network adapter 2"/>
                <vnsCIf name="Gig0/1" vnicName="Network adapter 3"/>
                <vnsCIf name="Gig0/2" vnicName="Network adapter 4"/>
                <vnsCIf name="Gig0/3" vnicName="Network adapter 5"/>
                <vnsCIf name="Gig0/4" vnicName="Network adapter 6"/>
                <vnsCIf name="Gig0/5" vnicName="Network adapter 7"/>
                <vnsCIf name="Gig0/6" vnicName="Network adapter 8"/>
                <vnsCIf name="Gig0/7" vnicName="Network adapter 9"/>
                <vnsCMgmt name="devMgmt" host="3.5.3.170" port="443"/>
                <vnsCCred name="username" value="admin"/>
                <vnsCCredSecret name="password" value="insieme"/>
            </vnsCDev>
        </vnsLDevVip>
    </fvTenant>
</polUni>
```

The following REST request creates a service graph in managed mode:

```
<polUni>
    <fvTenant name="acme">
        <vnsAbsGraph name = "G1">

        <vnsAbsTermNode name = "Input1">
            <vnsAbsTermConn name = "C1" direction = "output">
            </vnsAbsTermConn>
        </vnsAbsTermNode>

        <!-- Node1 Provides SLB functionality -->
        <vnsAbsNode name = "Node1" funcType="GoTo" >
            <vnsRsDefaultScopeToTerm
                tDn="uni/tn-acme/AbsGraph-G1/AbsTermNode-Output1/outtmnl"/>

            <vnsAbsFuncConn name = "C4" direction = "input">
               <vnsRsMConnAtt tDn="uni/infra/mDev-Acme-ADC-1.0/mFunc-SLB/mConn-external"/>

                <vnsRsConnToLIf tDn="uni/tn-acme/lDevVip-ADCCluster1/lIf-C4"/>
            </vnsAbsFuncConn>

            <vnsAbsFuncConn name = "C5" direction = "output">
               <vnsRsMConnAtt tDn="uni/infra/mDev-Acme-ADC-1.0/mFunc-SLB/mConn-internal"/>

                <vnsRsConnToLIf tDn="uni/tn-acme/lDevVip-ADCCluster1/lIf-C5"/>
            </vnsAbsFuncConn>

            <vnsRsNodeToMFunc tDn="uni/infra/mDev-Acme-ADC-1.0/mFunc-SLB"/>
        </vnsAbsNode>

        <vnsAbsTermNode name = "Output1">
            <vnsAbsTermConn name = "C6" direction = "input">
            </vnsAbsTermConn>
```

```
        </vnsAbsTermNode>

        <vnsAbsConnection name = "CON1">
            <vnsRsAbsConnectionConns
                tDn="uni/tn-acme/AbsGraph-G1/AbsTermNode-Input1/AbsTConn"/>
            <vnsRsAbsConnectionConns
                tDn="uni/tn-acme/AbsGraph-G1/AbsNode-Node1/AbsFConn-C4"/>
        </vnsAbsConnection>

        <vnsAbsConnection name = "CON3">
            <vnsRsAbsConnectionConns
                tDn="uni/tn-acme/AbsGraph-G1/AbsNode-Node1/AbsFConn-C5"/>
            <vnsRsAbsConnectionConns
                tDn="uni/tn-acme/AbsGraph-G1/AbsTermNode-Output1/AbsTConn"/>
        </vnsAbsConnection>
    </vnsAbsGraph>
  </fvTenant>
</polUni>
```

The following REST request creates a service graph in unmanaged mode:

```
<polUni>
    <fvTenant name="HA_Tenant1">
        <vnsAbsGraph name="g1">

            <vnsAbsTermNodeProv name="Input1">
                <vnsAbsTermConn name="C1">
                </vnsAbsTermConn>
            </vnsAbsTermNodeProv>

            <!-- Node1 Provides LoadBalancing functionality -->
            <vnsAbsNode name="Node1" managed="no">
                <vnsRsDefaultScopeToTerm
                  tDn="uni/tn-HA_Tenant1/AbsGraph-g1/AbsTermNodeProv-Input1/outtmnl"/>
                <vnsAbsFuncConn name="outside" attNotify="true">
                </vnsAbsFuncConn>
                <vnsAbsFuncConn name="inside" attNotify="true">
                </vnsAbsFuncConn>
            </vnsAbsNode>

            <vnsAbsTermNodeCon name="Output1">
                <vnsAbsTermConn name="C6">
                </vnsAbsTermConn>
            </vnsAbsTermNodeCon>

            <vnsAbsConnection name="CON2" adjType="L3" unicastRoute="yes">
                <vnsRsAbsConnectionConns
                  tDn="uni/tn-HA_Tenant1/AbsGraph-g1/AbsTermNodeCon-Output1/AbsTConn"/>
                <vnsRsAbsConnectionConns
                  tDn="uni/tn-HA_Tenant1/AbsGraph-g1/AbsNode-Node1/AbsFConn-outside"/>
            </vnsAbsConnection>

            <vnsAbsConnection name="CON1" adjType="L2" unicastRoute="no">
                <vnsRsAbsConnectionConns
                  tDn="uni/tn-HA_Tenant1/AbsGraph-g1/AbsNode-Node1/AbsFConn-inside"/>
                <vnsRsAbsConnectionConns
                  tDn="uni/tn-HA_Tenant1/AbsGraph-g1/AbsTermNodeProv-Input1/AbsTConn"/>
            </vnsAbsConnection>

        </vnsAbsGraph>
    </fvTenant>
</polUni>
```

The following REST request creates a filter and a security policy (contract):

```
<polUni>
    <fvTenant dn="uni/tn-acme" name="acme">
        <vzFilter name="HttpIn">
            <vzEntry name="e1" prot="6" dToPort="80"/>
        </vzFilter>

        <vzBrCP name="webCtrct">
            <vzSubj name="http">
                <vzRsSubjFiltAtt tnVzFilterName="HttpIn"/>
            </vzSubj>
        </vzBrCP>
    </fvTenant>
</polUni>
```

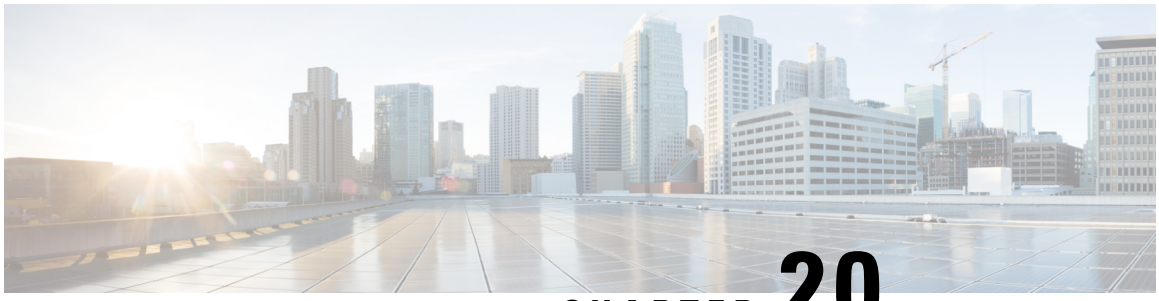
The following REST request provides graph configuration parameters from an application EPG:

```
<polUni>
   <fvTenant dn="uni/tn-acme" name="acme">

      <!-- Application Profile -->
      <fvAp dn="uni/tn-acme/ap-MyAP" name="MyAP">

          <!-- EPG 1  -->
          <fvAEPg dn="uni/tn-acme/ap-MyAP/epg-MyClnt" name="MyClnt">
              <fvRsBd tnFvBDName="MyClntBD"/>
              <fvRsDomAtt tDn="uni/vmmp-Vendor1/dom-MyVMs"/>
              <fvRsProv tnVzBrCPName="webCtrct">
              </fvRsProv>
            <fvRsPathAtt tDn="topology/pod-1/paths-17/pathep-[eth1/20]" encap="vlan-201"/>

              <fvSubnet name="SrcSubnet" ip="192.168.10.1/24"/>
          </fvAEPg>

          <!-- EPG 2 -->
          <fvAEPg dn="uni/tn-acme/ap-MyAP/epg-MySRVR"  name="MySRVR">
             <fvRsBd tnFvBDName="MyClntBD"/>
             <fvRsDomAtt tDn="uni/vmmp-Vendor1/dom-MyVMs"/>
             <fvRsCons tnVzBrCPName="webCtrct">
             </fvRsCons>

             <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any" nodeNameOrLbl="any"
               key="Monitor" name="monitor1">
                 <vnsParamInst name="weight" key="weight" value="10"/>
             </vnsFolderInst>

             <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any" nodeNameOrLbl="any"
               key="Service" name="Service1">
                 <vnsParamInst name="servicename" key="servicename"
                   value="crpvgrtst02-8010"/>
                 <vnsParamInst name="servicetype" key="servicetype" value="TCP"/>
                 <vnsParamInst name="servername" key="servername"
                   value="s192.168.100.100"/>
                 <vnsParamInst name="serveripaddress" key="serveripaddress"
                   value="192.168.100.100"/>
                 <vnsParamInst name="serviceport" key="serviceport" value="8080"/>
                 <vnsParamInst name="svrtimeout" key="svrtimeout" value="9000"/>
                 <vnsParamInst name="clttimeout" key="clttimeout" value="9000"/>
                 <vnsParamInst name="usip" key="usip" value="NO"/>
                 <vnsParamInst name="useproxyport" key="useproxyport" value=""/>
                 <vnsParamInst name="cip" key="cip" value="ENABLED"/>
                 <vnsParamInst name="cka" key="cka" value="NO"/>
                 <vnsParamInst name="sp" key="sp" value="OFF"/>
                 <vnsParamInst name="cmp" key="cmp" value="NO"/>
                 <vnsParamInst name="maxclient" key="maxclient" value="0"/>
                 <vnsParamInst name="maxreq" key="maxreq" value="0"/>
```

```
                    <vnsParamInst name="tcpb" key="tcpb" value="NO"/>
                    <vnsCfgRelInst name="MonitorConfig" key="MonitorConfig"
                      targetName="monitor1"/>
                  </vnsFolderInst>

                  <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any"
                    nodeNameOrLbl="any" key="Network" name="Network">
                    <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any"
                      nodeNameOrLbl="any" key="vip" name="vip">
                        <vnsParamInst name="vipaddress1" key="vipaddress"
                          value="10.10.10.100"/>
                    </vnsFolderInst>
                    <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any"
                      nodeNameOrLbl="any" devCtxLbl="C1" key="snip" name="snip1">
                        <vnsParamInst name="snipaddress" key="snipaddress"
                          value="192.168.1.100"/>
                    </vnsFolderInst>
                    <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any"
                      nodeNameOrLbl="any" devCtxLbl="C2" key="snip" name="snip2">
                        <vnsParamInst name="snipaddress" key="snipaddress"
                          value="192.168.1.101"/>
                    </vnsFolderInst>
                    <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any"
                      nodeNameOrLbl="any" devCtxLbl="C3" key="snip" name="snip3">
                        <vnsParamInst name="snipaddress" key="snipaddress"
                          value="192.168.1.102"/>
                    </vnsFolderInst>
                  </vnsFolderInst>

                  <!-- SLB Configuration -->
                  <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any"
                   nodeNameOrLbl="any" key="VServer" name="VServer">
                    <!-- Virtual Server Configuration -->
                    <vnsParamInst name="port" key="port" value="8010"/>
                    <vnsParamInst name="vip" key="vip" value="10.10.10.100"/>
                    <vnsParamInst name="vservername" key="vservername"
                      value="crpvgrtst02-vip-8010"/>
                    <vnsParamInst name="servicename" key="servicename"
                      value="crpvgrtst02-8010"/>
                    <vnsParamInst name="servicetype" key="servicetype" value="TCP"/>
                    <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any"
                    nodeNameOrLbl="any" key="VServerGlobalConfig" name="VServerGlobalConfig">

                        <vnsCfgRelInst name="ServiceConfig" key="ServiceConfig"
                          targetName="Service1"/>
                        <vnsCfgRelInst name="VipConfig" key="VipConfig"
                          targetName="Network/vip"/>
                    </vnsFolderInst>
                  </vnsFolderInst>
              </fvAEPg>
          </fvAp>
      </fvTenant>
</polUni>
```

The following REST request attaches a service graph to a contract:

```
<polUni>
    <fvTenant name="acme">
        <vzBrCP name="webCtrct">
            <vzSubj name="http">
                <vzRsSubjGraphAtt graphName="G1" termNodeName="Input1"/>
            </vzSubj>
        </vzBrCP>
    </fvTenant>
</polUni>
```



CHAPTER 20

Configuring Cloud Orchestrator Mode

- About the Cloud Orchestrator Mode, on page 197
- Cloud Orchestrator Mode Schema, on page 197
- Configuring the Cloud Orchestrator Mode Using the GUI, on page 204
- Configuring a Firewall Using the REST API, on page 205
- Configuring a Load Balancer Using the REST API, on page 206

About the Cloud Orchestrator Mode

In environments where the Cisco APIC works with a cloud Orchestrator, such as Azure, vRealize, OpenStack, or CliQR, the cloud Orchestrator must typically be aware of the semantics of the vendor's configuration parameters. With the cloud orchestrator mode, however, the Cisco APIC provides an LB-aas and a FW-aas interface to enable a standard set of parameters that creates a unified interface for configuring load balancers and firewalls in a service graph. The cloud orchestrator mode can also work with the unified interface to provision load balancers and firewalls in the Cisco ACI fabric. As a result, the Orchestrator does not need to be aware of the semantics of the vendor's configuration parameters.

Cloud Orchestrator Mode Schema

Firewall Schema

The cloud orchestrator mode schema is published as a device package (CISCO CloudMode device package) that is automatically created in the Cisco APIC.

Figure 40: Firewall Interface

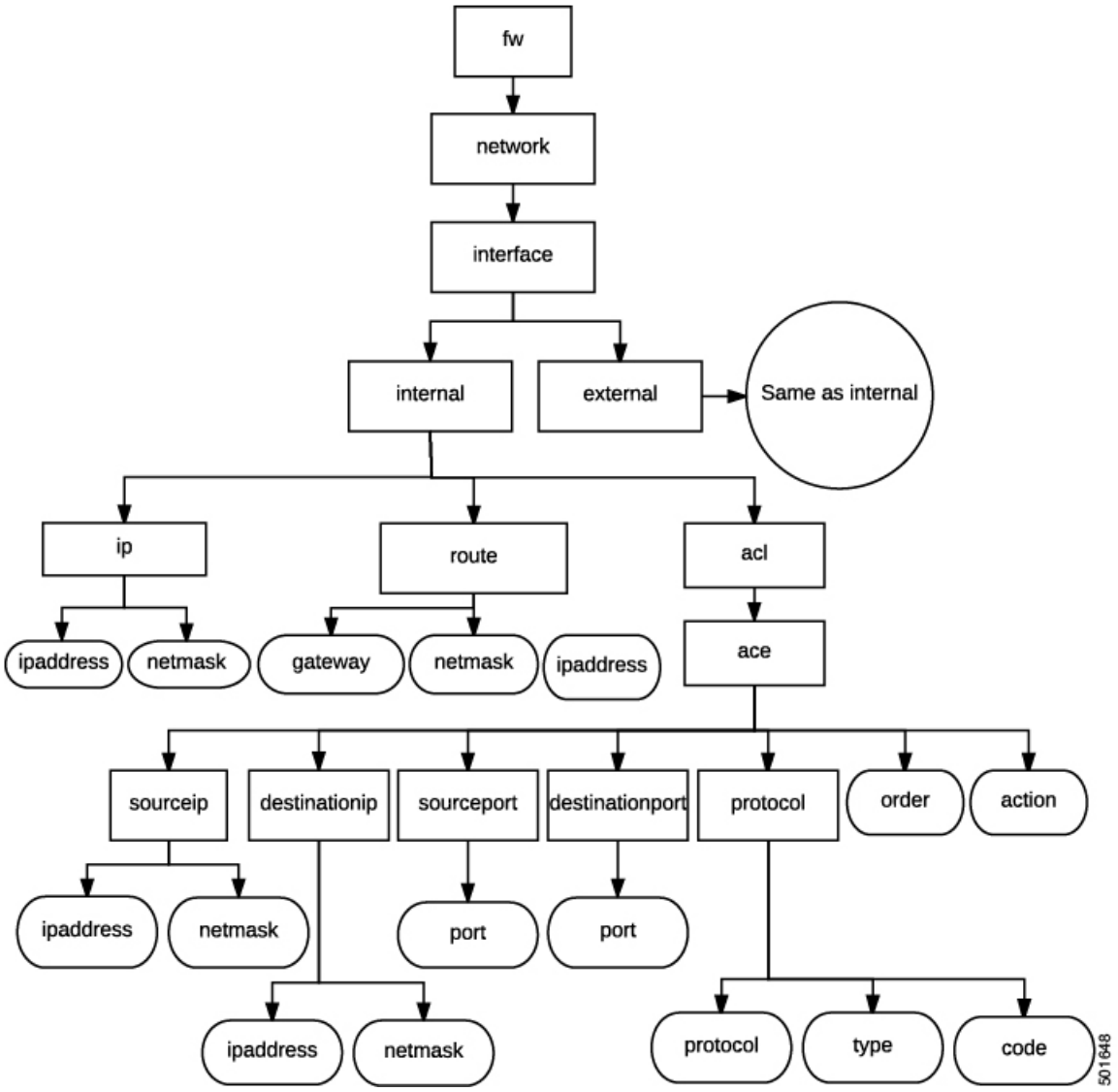


Figure 41: Firewall NAT

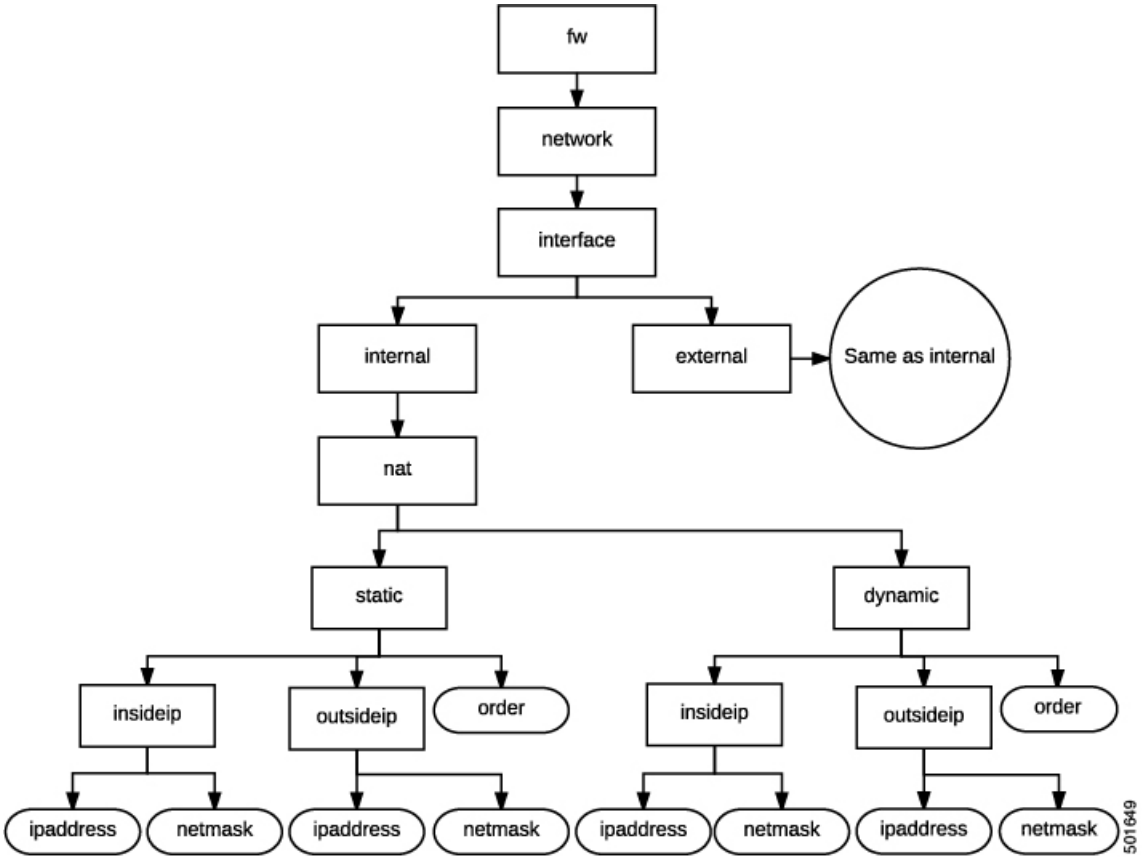


Figure 42: Firewall PAT

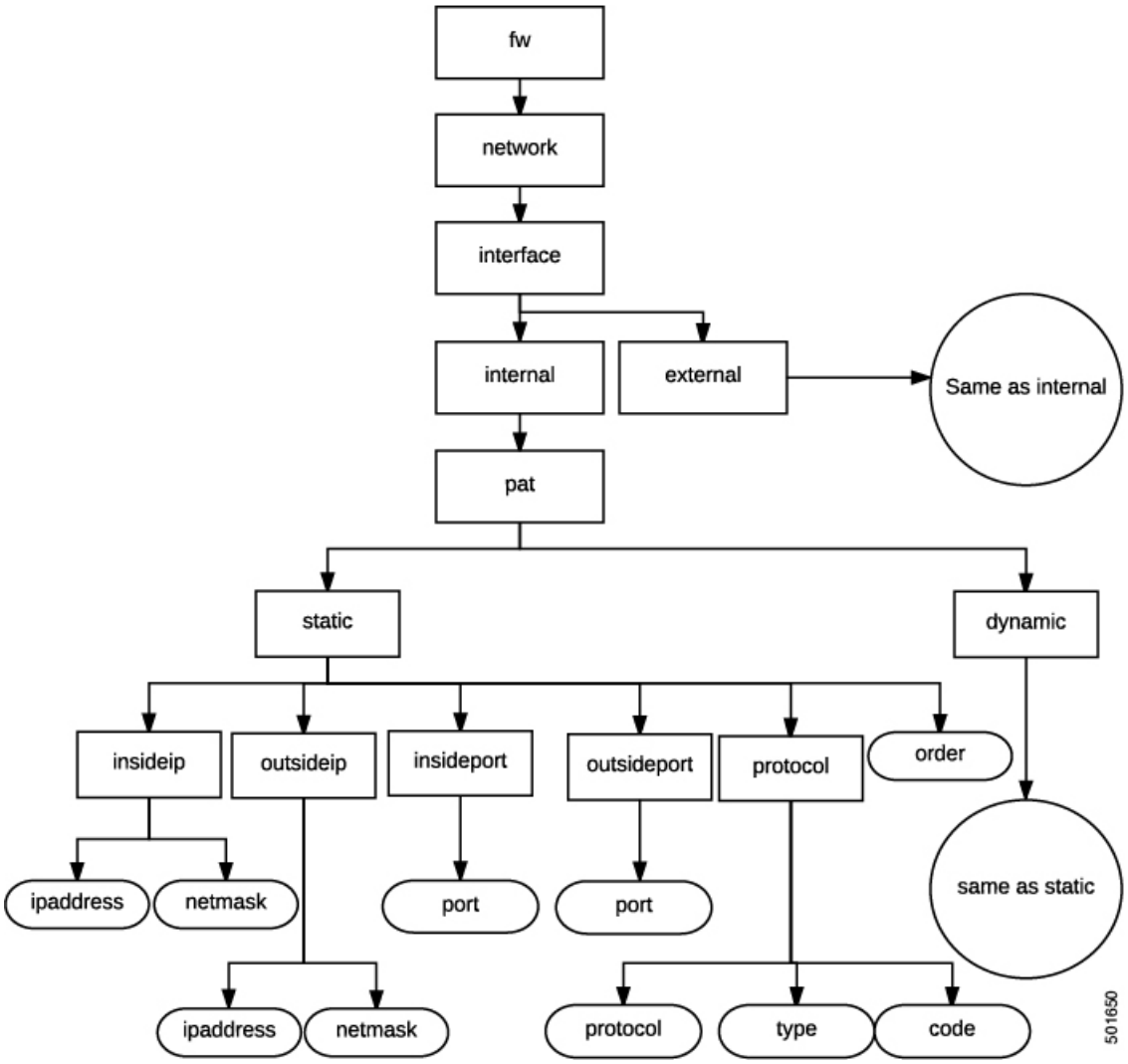
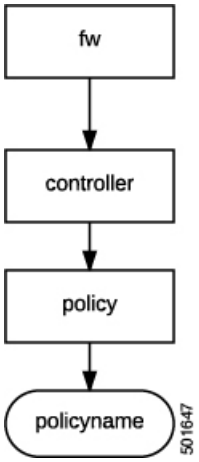


Figure 43: Firewall Controller



Load Balancer Schema

The cloud orchestrator mode schema is published as a device package (CISCO CloudMode device package) that is automatically created in the Cisco APIC.

Figure 44: Load Balancer Interface

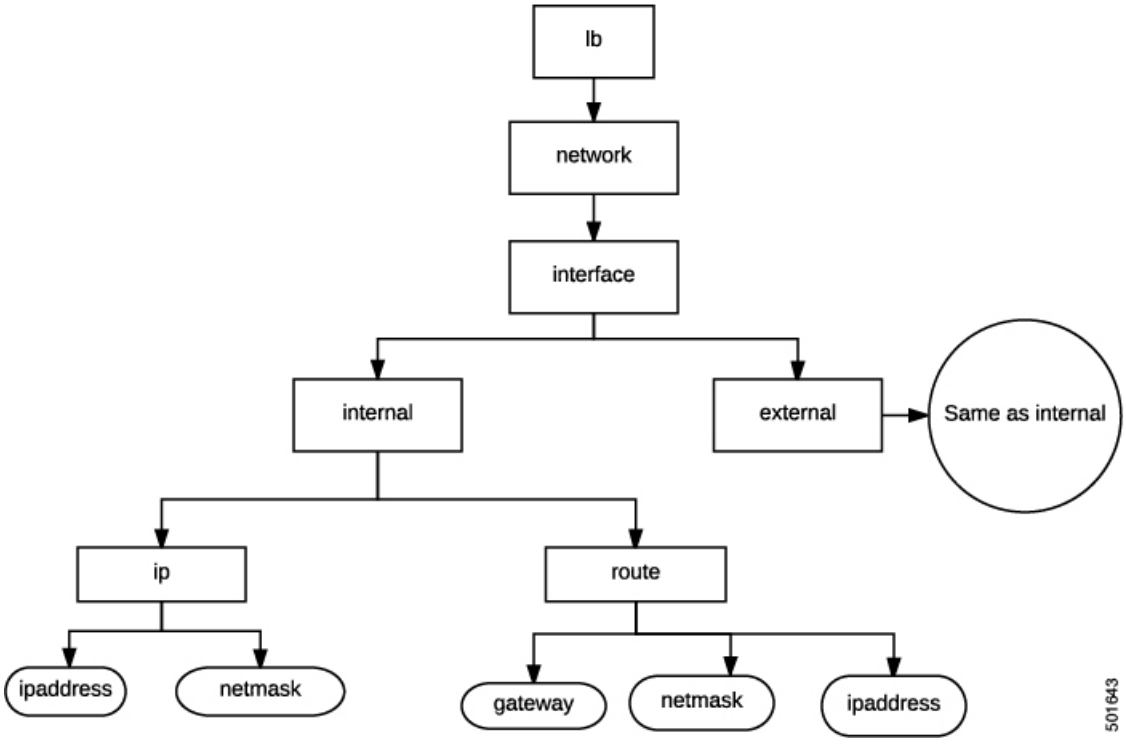


Figure 45: Load Balancer NAT

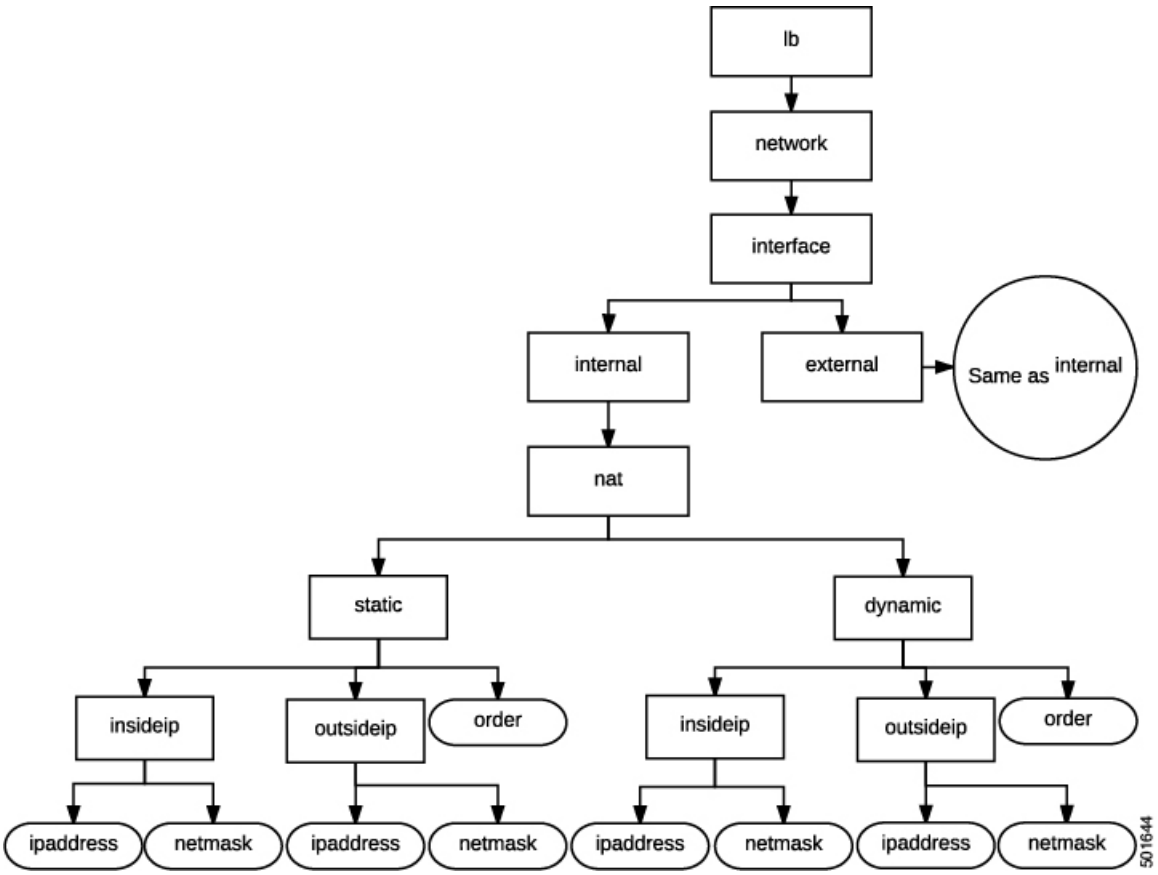


Figure 46: Load Balancer PAT

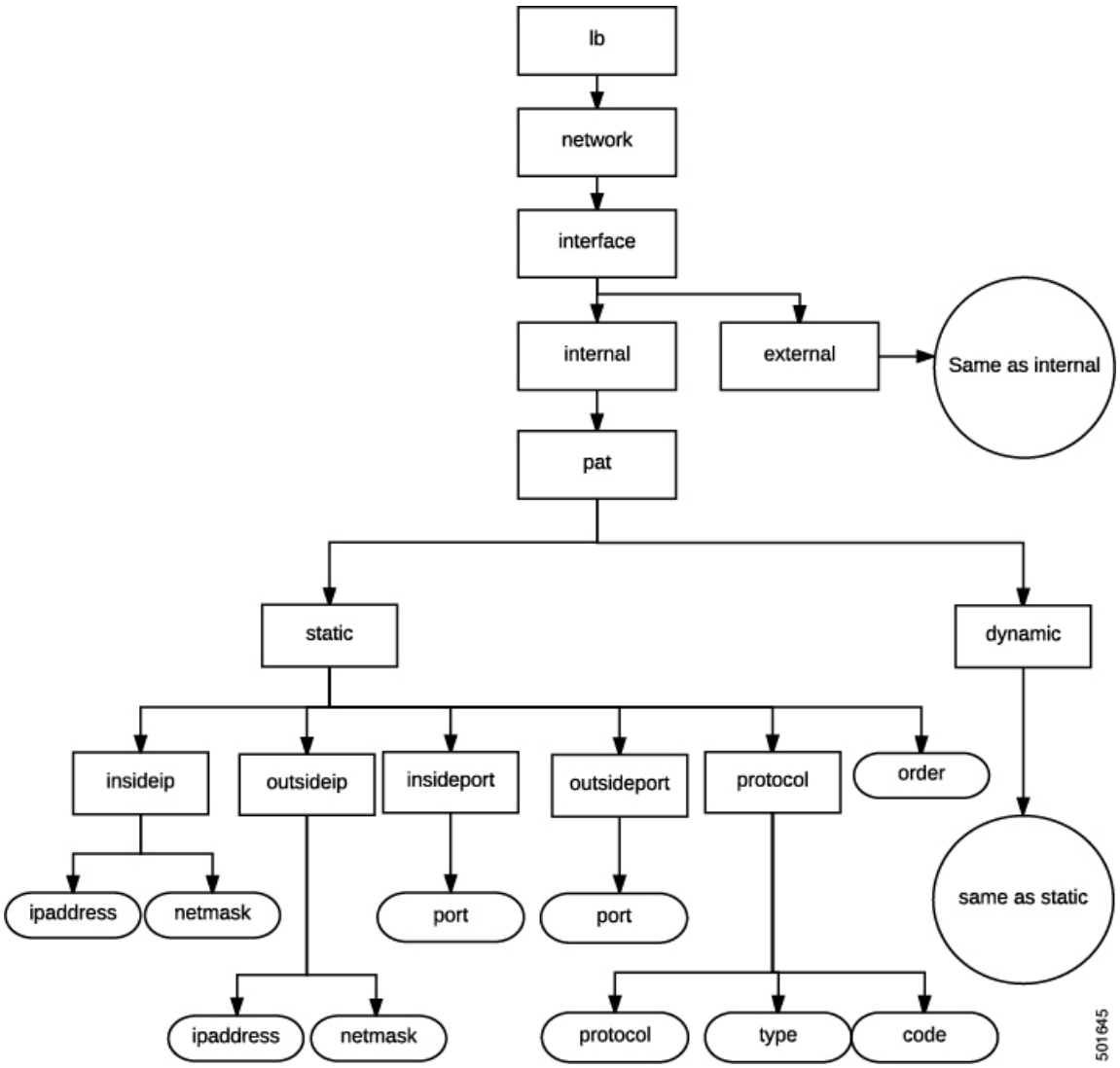


Figure 47: Load Balancer Controller

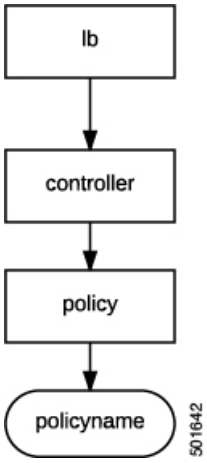
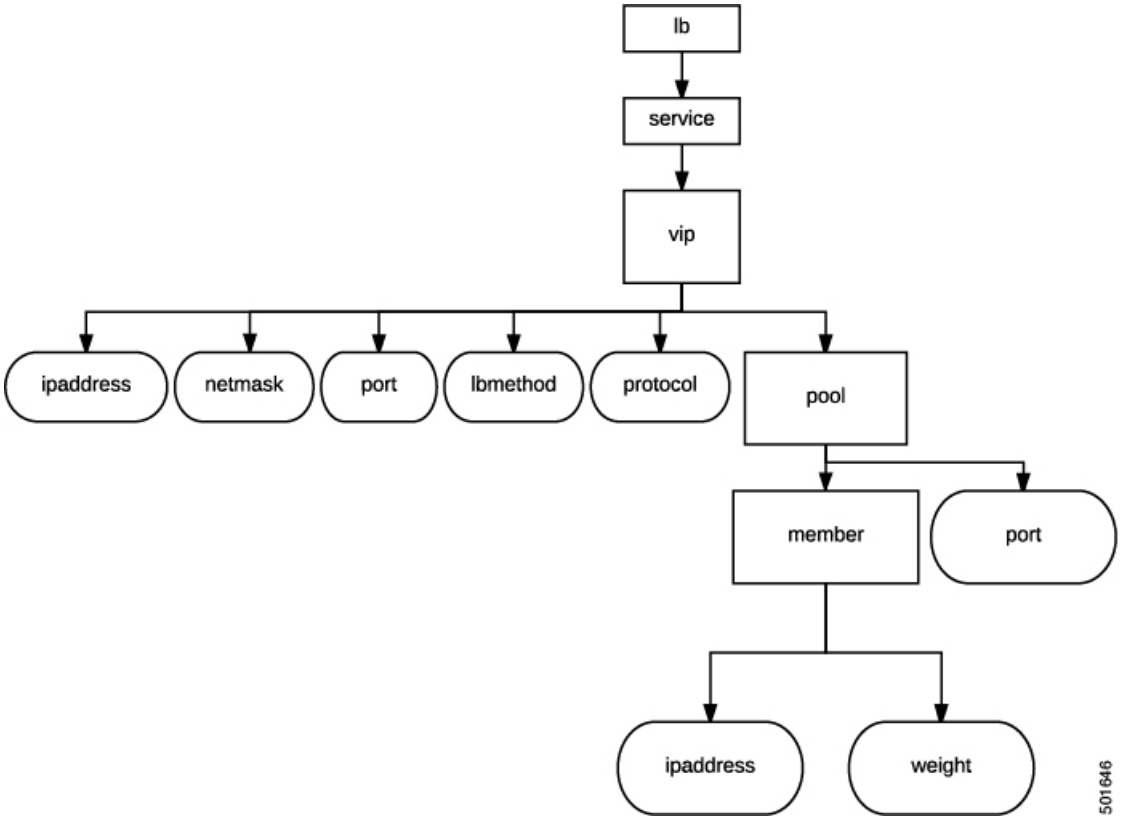


Figure 48: Load Balancer Service



Configuring the Cloud Orchestrator Mode Using the GUI

The cloud orchestrator mode is configured in the GUI when performing the following actions:

- **Creating a Function Profile**–When creating a Layer 4 to Layer 7 function profile, the option to use the cloud orchestrator mode appears when choosing an existing profile. The **Profile** property appears as a drop-down menu that lists the supported function profiles. The cloud orchestrator mode profiles are identified in the profile name.

Note Device packages from vendors using cloud orchestrator mode will have pre-created copies of function profiles in cloud mode for all of their existing function profiles.

For information about creating a function profile, see Creating a Function Profile Using the GUI, on page 46.

- **Creating a Layer 4 to Layer 7 Service Graph Template**–When creating a Layer 4 to Layer 7 service graph template, the option to use the cloud orchestrator mode appears when creating a service node. The **Profile** property appears as a drop-down menu that lists the supported profiles. The cloud orchestrator mode profiles are identified in the profile name.

  For information about creating a service node when creating a Layer 4 to Layer 7 Service Graph Template, see Configuring a Service Graph Template Using the GUI, on page 38 .

- **Applying a Service Graph Template to Endpoint Groups**–The cloud orchestrator mode interface for the chosen profile appears when applying a service graph template to EPGs with a cloud orchestrator mode profile.

  For more information about applying a service graph template to EPGs, see Applying a Service Graph Template to Endpoint Groups Using the GUI, on page 41.

Configuring a Firewall Using the REST API

The following REST API configures a firewall:

```
<fvTenant  name="Tenant1">
  <fvAp name="ap1">
   <fvAEPg name="epg3">
    <vnsSvcPol ctrct="ctrct_fw" graph="Graph_FW" node="FW">
     <vnsFolderInst ctrctNameOrLbl="ctrct_fw" graphNameOrLbl="Graph_FW" nodeNameOrLbl="FW"
 key="fw" name="fw">
     <vnsRsFolderInstToMFolder tDn="uni/infra/mDev-CISCO-CloudMode-1.0/mFunc-FW/mFolder-fw"/>

      <vnsFolderInst ctrctNameOrLbl="ctrct_fw" graphNameOrLbl="Graph_FW" nodeNameOrLbl="FW"
 key="network" name="network">
       <vnsFolderInst ctrctNameOrLbl="ctrct_fw" graphNameOrLbl="Graph_FW" nodeNameOrLbl="FW"
 key="interface" name="interface">
        <vnsFolderInst ctrctNameOrLbl="ctrct_fw" graphNameOrLbl="Graph_FW" nodeNameOrLbl="FW"
 key="internal" name="internal">
         <vnsFolderInst ctrctNameOrLbl="ctrct_fw" graphNameOrLbl="Graph_FW" nodeNameOrLbl="FW"
 key="ip" name="ip">
           <vnsParamInst name="ipaddress" key="ipaddress" value="2.2.2.2"/>
           <vnsParamInst name="netmask" key="netmask" value="255.255.255.0"/>
          </vnsFolderInst>
         </vnsFolderInst>
        <vnsFolderInst ctrctNameOrLbl="ctrct_fw" graphNameOrLbl="Graph_FW" nodeNameOrLbl="FW"
 key="external" name="external">
         <vnsFolderInst ctrctNameOrLbl="ctrct_fw" graphNameOrLbl="Graph_FW" nodeNameOrLbl="FW"
 key="ip" name="ip">
           <vnsParamInst name="ipaddress" key="ipaddress" value="1.1.1.1"/>
```

```
          <vnsParamInst name="netmask" key="netmask" value="255.255.255.0"/>
         </vnsFolderInst>
        <vnsFolderInst ctrctNameOrLbl="ctrct_fw" graphNameOrLbl="Graph_FW" nodeNameOrLbl="FW"
 key="acl" name="acl">
          <vnsFolderInst ctrctNameOrLbl="ctrct_fw" graphNameOrLbl="Graph_FW"
nodeNameOrLbl="FW" key="ace" name="ace">
           <vnsParamInst name="action" key="action" value="PERMIT"/>
           <vnsParamInst name="order" key="order" value="10"/>
           <vnsFolderInst ctrctNameOrLbl="ctrct_fw" graphNameOrLbl="Graph_FW"
nodeNameOrLbl="FW" key="protocol" name="protocol">
            <vnsParamInst name="protocol" key="protocol" value="TCP"/>
           </vnsFolderInst>
           <vnsFolderInst ctrctNameOrLbl="ctrct_fw" graphNameOrLbl="Graph_FW"
nodeNameOrLbl="FW" key="sourceip" name="sourceip">
            <vnsParamInst name="ipaddress" key="ipaddress" value="0.0.0.0"/>
            <vnsParamInst name="netmask" key="netmask" value="0"/>
           </vnsFolderInst>
           <vnsFolderInst ctrctNameOrLbl="ctrct_fw" graphNameOrLbl="Graph_FW"
nodeNameOrLbl="FW" key="destinationip" name="destinationip">
            <vnsParamInst name="ipaddress" key="ipaddress" value="10.10.10.0"/>
            <vnsParamInst name="netmask" key="netmask" value="24"/>
           </vnsFolderInst>
           <vnsFolderInst ctrctNameOrLbl="ctrct_fw" graphNameOrLbl="Graph_FW"
nodeNameOrLbl="FW" key="destinationport" name="destinationport">
            <vnsParamInst name="port" key="port" value="80"/>
           </vnsFolderInst>
          </vnsFolderInst>
         </vnsFolderInst>
        </vnsFolderInst>
       </vnsFolderInst>
      </vnsFolderInst>
     </vnsFolderInst>
    </vnsSvcPol>
   </fvAEPg>
  </fvAp>
</fvTenant>
```

Configuring a Load Balancer Using the REST API

The following REST API configures a load balancer:

```
<fvTenant  name="Tenant1">
  <fvAp name="ap1">
   <fvAEPg name="epg2">
    <vnsSvcPol ctrct="ctrct_lb" graph="Graph_ADC" node="ADC">
     <vnsFolderInst ctrctNameOrLbl="ctrct_lb" graphNameOrLbl="Graph_ADC" nodeNameOrLbl="ADC"
 key="lb" name="lb">
     <vnsRsFolderInstToMFolder tDn="uni/infra/mDev-CISCO-CloudMode-1.0/mFunc-LB/mFolder-lb"/>

     <vnsFolderInst ctrctNameOrLbl="ctrct_lb" graphNameOrLbl="Graph_ADC" nodeNameOrLbl="ADC"
 key="network" name="network">
      <vnsFolderInst ctrctNameOrLbl="ctrct_lb" graphNameOrLbl="Graph_ADC" nodeNameOrLbl="ADC"
 key="interface" name="interface">
       <vnsFolderInst ctrctNameOrLbl="ctrct_lb" graphNameOrLbl="Graph_ADC"
nodeNameOrLbl="ADC" key="internal" name="internal">
        <vnsFolderInst ctrctNameOrLbl="ctrct_lb" graphNameOrLbl="Graph_ADC"
nodeNameOrLbl="ADC" key="ip" name="ip">
         <vnsParamInst name="ipaddress" key="ipaddress" value="2.2.2.2"/>
         <vnsParamInst name="netmask" key="netmask" value="255.255.255.0"/>
        </vnsFolderInst>
       </vnsFolderInst>
       <vnsFolderInst ctrctNameOrLbl="ctrct_lb" graphNameOrLbl="Graph_ADC"
```

```
nodeNameOrLbl="ADC" key="external" name="external">
         <vnsFolderInst ctrctNameOrLbl="ctrct_lb" graphNameOrLbl="Graph_ADC"
nodeNameOrLbl="ADC" key="ip" name="ip">
          <vnsParamInst name="ipaddress" key="ipaddress" value="1.1.1.1"/>
          <vnsParamInst name="netmask" key="netmask" value="255.255.255.0"/>
         </vnsFolderInst>
        </vnsFolderInst>
       </vnsFolderInst>
      </vnsFolderInst>
     <vnsFolderInst ctrctNameOrLbl="ctrct_lb" graphNameOrLbl="Graph_ADC" nodeNameOrLbl="ADC"
 key="service" name="service">
      <vnsFolderInst ctrctNameOrLbl="ctrct_lb" graphNameOrLbl="Graph_ADC" nodeNameOrLbl="ADC"
 key="vip" name="vip1">
        <vnsParamInst name="lbmethod" key="lbmethod" value="LEAST_CONNECTIONS"/>
        <vnsParamInst name="protocol" key="protocol" value="TCP"/>
        <vnsParamInst name="ipaddress" key="ipaddress" value="3.3.3.3"/>
        <vnsParamInst name="port" key="port" value="80"/>
        <vnsFolderInst ctrctNameOrLbl="ctrct_lb" graphNameOrLbl="Graph_ADC"
nodeNameOrLbl="ADC" key="pool" name="pool1">
         <vnsParamInst name="port" key="port" value="80"/>
        </vnsFolderInst>
       </vnsFolderInst>
      </vnsFolderInst>
     </vnsFolderInst>
    </vnsSvcPol>
   </fvAEPg>
  </fvAp>
</fvTenant>
```