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.

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: http://www.cisco.com/go/trademarks. 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. (1110R)

© 2016–2019 Cisco Systems, Inc. All rights reserved.
CONTENTS

PREFACE

Preface xi
  Audience xi
  Document Conventions xi
  Related Documentation xiii
  Documentation Feedback xiii
  Obtaining Documentation and Submitting a Service Request xiii

CHAPTER 1

New and Changed Information 1
  New and Changed Information 1

CHAPTER 2

Overview 5
  About Deploying Application-Centric Infrastructure Layer 4 to Layer 7 Services 5
  Configuring Layer 4 to Layer 7 Services Using the GUI 6
  About Service Graph Templates 6

CHAPTER 3

Importing a Device Package 9
  About Device Packages 9
  Installing a Device Package Using the REST API 11
  Importing a Device Package Using the GUI 12

CHAPTER 4

Defining a Logical Device 13
  About Device Clusters 13
    About Managed Device Clusters 14
    About Unmanaged Device Clusters 14
  About Concrete Devices 14
  About Trunking 15
CHAPTER 10

Configuring Direct Server Return 107
About Direct Server Return 107
Layer 2 Direct Server Return 108
About Deploying Layer 2 Direct Server Return with Cisco Application Centric Infrastructure 109
Guidelines and Limitations for Configuring Direct Server Return 110
Supported Direct Server Return Configuration 111
Example XML POST of Direct Server Return for Static Service Deployment 111
Direct Server Return for Static Service Deployment 112
Direct Server Return for Static Service Deployment Logical Model 112
Direct Server Return for Service Graph Insertion 112
Direct Server Return Shared Layer 4 to Layer 7 Service Configuration 113
Configuring the Citrix Server Load Balancer for Direct Server Return 113
Configuring a Linux Server for Direct Server Return 114
CHAPTER 11

Configuring the Device and Chassis Manager  115
  About Device Managers and Chassis Managers  115
  Device Manager and Chassis Manager Behavior  118
  Creating Devices Using the GUI  119
  Creating a Chassis Using the GUI  119
  Example XML for Device Managers and Chassis Managers  120
    Example XML for Creating the MDevMgr Object  120
    Example XML for Associating an LDevVip Object With a DevMgr Object  120
    Example XML for Creating the MChassis Object  121
    Example XML for Creating the Chassis Object  121
    Example XML for Associating an CDev Object With a Chassis Object  121
  Device and Chassis Callouts  122
    Example deviceValidate Callout for a Device  122
    Example deviceAudit Callout for a Device  122
    Example clusterAudit Callout for a Device  122
    Example serviceAudit Callout for a Device  123
    Example deviceValidate Callout for a Chassis  124
    Example deviceAudit Callout for a Chassis  124
    Example clusterAudit Callout for a Chassis  124
    Example serviceAudit Callout for a Chassis  125

CHAPTER 12

Configuring Unmanaged Mode  127
  About the Unmanaged Mode  127
  About Managed and Unmanaged Logical Devices  127
  About Managed and Unmanaged Function Nodes  128
  About Layer 4 to Layer 7 Services Endpoint Groups  129
  Using Static Encapsulation for a Graph Connector  129
  Creating a Physical Device Using the NX-OS-Style CLI  129
  Creating a High Availability Cluster Using the NX-OS-Style CLI  131
  Creating a Virtual Device Using the NX-OS-Style CLI  132
  Example XML for the Unmanaged Mode  133
    Example XML of Creating an Unmanaged LDevVip Object  133
    Example XML of Creating an Unmanaged AbsNode Object  133
Preface

This preface includes the following sections:

- Audience, on page xi
- Document Conventions, on page xi
- Related Documentation, on page xiii
- Documentation Feedback, on page xiii
- Obtaining Documentation and Submitting a Service Request, on page xiii

Audience

This guide is intended primarily for data center administrators with responsibilities and expertise in one or more of the following:

- Virtual machine installation and administration
- Layer 4 to Layer 7 Services installation and administration
- Switch and network administration

Document Conventions

Command descriptions use the following conventions:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>bold</strong></td>
<td>Bold text indicates the commands and keywords that you enter literally as shown.</td>
</tr>
<tr>
<td><em>Italic</em></td>
<td>Italic text indicates arguments for which the user supplies the values.</td>
</tr>
<tr>
<td>[x]</td>
<td>Square brackets enclose an optional element (keyword or argument).</td>
</tr>
<tr>
<td>[x</td>
<td>y]</td>
</tr>
<tr>
<td>{x</td>
<td>y}</td>
</tr>
<tr>
<td>Convention</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>[x {y</td>
<td>z}]</td>
</tr>
<tr>
<td>variable</td>
<td>Indicates a variable for which you supply values, in context where italics cannot be used.</td>
</tr>
<tr>
<td>string</td>
<td>A nonquoted set of characters. Do not use quotation marks around the string or the string will include the quotation marks.</td>
</tr>
</tbody>
</table>

Examples use the following conventions:

<table>
<thead>
<tr>
<th>Convention</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>screen font</strong></td>
<td>Terminal sessions and information the switch displays are in screen font.</td>
</tr>
<tr>
<td><strong>boldface screen font</strong></td>
<td>Information you must enter is in boldface screen font.</td>
</tr>
<tr>
<td><strong>italic screen font</strong></td>
<td>Arguments for which you supply values are in italic screen font.</td>
</tr>
<tr>
<td>&lt;&gt;</td>
<td>Nonprinting characters, such as passwords, are in angle brackets.</td>
</tr>
<tr>
<td>[ ]</td>
<td>Default responses to system prompts are in square brackets.</td>
</tr>
<tr>
<td>!, #</td>
<td>An exclamation point (!) or a pound sign (#) at the beginning of a line of code indicates a comment line.</td>
</tr>
</tbody>
</table>

This document uses the following conventions:

**Note**

Means *reader take note*. Notes contain helpful suggestions or references to material not covered in the manual.

**Caution**

Means *reader be careful*. In this situation, you might do something that could result in equipment damage or loss of data.

**Warning**

IMPORTANT SAFETY INSTRUCTIONS

This warning symbol means danger. You are in a situation that could cause bodily injury. Before you work on any equipment, be aware of the hazards involved with electrical circuitry and be familiar with standard practices for preventing accidents. Use the statement number provided at the end of each warning to locate its translation in the translated safety warnings that accompanied this device.

SAVE THESE INSTRUCTIONS
Related Documentation

Cisco Application Centric Infrastructure (ACI) Documentation

Cisco Application Centric Infrastructure (ACI) Simulator Documentation

Cisco Nexus 9000 Series Switches Documentation

Cisco Application Virtual Switch Documentation

Cisco Application Centric Infrastructure (ACI) Integration with OpenStack Documentation

Documentation Feedback
To provide technical feedback on this document, or to report an error or omission, please send your comments to apic-docfeedback@cisco.com. We appreciate your feedback.

Obtaining Documentation and Submitting a Service Request
For information on obtaining documentation, using the Cisco Bug Search Tool (BST), submitting a service request, and gathering additional information, see What’s New in Cisco Product Documentation at: http://www.cisco.com/c/en/us/td/docs/general/whatsnew/whatsnew.html
Subscribe to What’s New in Cisco Product Documentation, which lists all new and revised Cisco technical documentation as an RSS feed and delivers content directly to your desktop using a reader application. The RSS feeds are a free service.
New and Changed Information

This chapter contains the following sections:

• 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 2.0(1).

<table>
<thead>
<tr>
<th>Cisco APIC Release Version</th>
<th>Feature</th>
<th>Description</th>
<th>Where Documented</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0(1)</td>
<td>Copy Services</td>
<td>Unlike SPAN that duplicates all the traffic, the Cisco Application Centric Infrastructure (ACI) contract copy 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 plan traffic that are not covered by the contract are not copied. SPAN copies everything out of endpoint groups, access ports or uplink ports. Unlike SPAN, copy contracts do not add headers to the copied traffic. Copy contract traffic is managed internally in the switch to minimize impact on normal traffic forwarding.</td>
<td>Configuring Copy Services, on page 137</td>
</tr>
</tbody>
</table>
### Policy-Based Redirect

**Description**: 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 all to be in the same VRF instance.

**Note**: The GUI refers to this feature as "policy-based routing".

<table>
<thead>
<tr>
<th>Cisco APIC Release Version</th>
<th>Feature</th>
<th>Description</th>
<th>Where Documented</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.0(1)</td>
<td>Policy-Based Redirect</td>
<td>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 all to be in the same VRF instance. Note: The GUI refers to this feature as &quot;policy-based routing&quot;.</td>
<td>Configuring Policy-Based Redirect, on page 65</td>
</tr>
<tr>
<td>1.3(1)</td>
<td>Object Model CLI</td>
<td>Removed all object model CLI procedures. The object model CLI is being deprecated in the next major release. Use the NX-OS-style CLI instead.</td>
<td>-</td>
</tr>
</tbody>
</table>
Overview

- About Deploying Application-Centric Infrastructure Layer 4 to Layer 7 Services, on page 5
- Configuring Layer 4 to Layer 7 Services Using the GUI, on page 6
- About Service Graph Templates, on page 6

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.
   1. Select the default service graph template parameters from an application profile.
   2. 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.

---

**Configuring Layer 4 to Layer 7 Services Using the GUI**

Using the GUI, you can configure the Layer 4 to layer 7 services for the Application Policy Infrastructure Controller (APIC).

See **Using the GUI, on page 193** for the procedures for configuring the services and service graph templates.

**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 a 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 the 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.

The 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 graph templates identify the set of network or service functions that are needed by the application. Once 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 template. The 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.
Importing a Device Package

• About Device Packages, on page 9
• Installing a Device Package Using the REST API, on page 11
• Importing a Device Package Using the GUI, on page 12

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 APIC through the device package. A device package manages a single class of service devices and provides the APIC with information about the device and its capabilities. A device package is a zip file that contains the following parts:

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

| Device script | A Python script that interacts with the device from the APIC. 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. |
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 APIC:

*Figure 1: Device Package and the 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 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 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**

You can import a device package using the GUI.

See *Using the GUI, on page 193* for the procedure for importing a device package.
Defining a Logical Device

- About Device Clusters, on page 13
- About Concrete Devices, on page 14
- About Trunking, on page 15
- Creating a Layer 4 to Layer 7 Device Using the GUI, on page 15
- Creating a Layer 4 to Layer 7 Device Using the NX-OS-Style CLI, on page 18
- Enabling Trunking on a Layer 4 to Layer 7 Virtual ASA device Using the GUI, on page 22
- Enabling Trunking on a Layer 4 to Layer 7 Virtual ASA device Using the REST APIs, on page 22
- Using an Imported Device with the REST APIs, on page 23
- Importing a Device From Another Tenant Using the NX-OS-Style CLI, on page 23
- Verifying the Import of a Device Using the GUI, on page 24

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.
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.

**Creating a Layer 4 to Layer 7 Device Using the GUI**

When you create a Layer 4 to Layer 7 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 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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Managed check box</td>
<td>Put a check in the box to create a managed device, or remove the check from the box to create an unmanaged device.</td>
</tr>
<tr>
<td>Name field</td>
<td>Enter a name for the device.</td>
</tr>
</tbody>
</table>
### Defining a Logical Device

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Type</td>
<td>Choose the service type. The types are:</td>
</tr>
<tr>
<td></td>
<td>• ADC</td>
</tr>
<tr>
<td></td>
<td>• Firewall</td>
</tr>
<tr>
<td></td>
<td>• Other</td>
</tr>
<tr>
<td>Note</td>
<td>For L1/L2 Firewall configuration use <strong>Other</strong>.</td>
</tr>
<tr>
<td>Device Type buttons</td>
<td>Choose the device type.</td>
</tr>
<tr>
<td>Physical Domain</td>
<td>Choose the physical domain or VMM domain.</td>
</tr>
<tr>
<td>or VMM Domain</td>
<td></td>
</tr>
<tr>
<td>Switching Mode</td>
<td>For a Cisco ACI Virtual Edge virtual domain, choose one of the following modes:</td>
</tr>
<tr>
<td>(Cisco ACI</td>
<td>• <strong>AVE</strong>—Traffic is switched through the Cisco ACI Virtual Edge.</td>
</tr>
<tr>
<td>Virtual Edge only)</td>
<td>• <strong>native</strong>—Traffic is switched through the VMware DVS.</td>
</tr>
<tr>
<td>View radio buttons</td>
<td>Choose the view for the device. The view can be:</td>
</tr>
<tr>
<td></td>
<td>• <strong>Single Node</strong>—Only one node</td>
</tr>
<tr>
<td></td>
<td>• <strong>HA Node</strong>—High availability nodes (two nodes)</td>
</tr>
<tr>
<td></td>
<td>• <strong>Cluster</strong>—3 or more nodes</td>
</tr>
<tr>
<td>Device Package</td>
<td>(Only for managed devices) Choose the vendor-provided device package that you will use.</td>
</tr>
<tr>
<td>drop-down list</td>
<td></td>
</tr>
<tr>
<td>Model drop-down list</td>
<td>(Only for managed devices) Choose the model of the device.</td>
</tr>
<tr>
<td>Context Aware</td>
<td>The context awareness of the device. The awareness can be:</td>
</tr>
<tr>
<td></td>
<td>• <strong>Single</strong>: 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.</td>
</tr>
<tr>
<td></td>
<td>• <strong>Multiple</strong>: 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.</td>
</tr>
<tr>
<td></td>
<td>The default is <strong>Single</strong>.</td>
</tr>
<tr>
<td>Note</td>
<td>When you create a Layer 4 to Layer 7 service device that is a load balancer, the <strong>Context Aware</strong> parameter is not used and can be ignored.</td>
</tr>
</tbody>
</table>
Defining a Logical Device

Creating a Layer 4 to Layer 7 Device Using the GUI

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Function Type</td>
<td>Function types are:</td>
</tr>
<tr>
<td></td>
<td>• GoThrough: Transparent mode</td>
</tr>
<tr>
<td></td>
<td>• GoTo: Routed mode</td>
</tr>
<tr>
<td></td>
<td>• L1: L1 Firewall mode</td>
</tr>
<tr>
<td></td>
<td>• L2: L2 Firewall mode</td>
</tr>
</tbody>
</table>

The default is GoTo.

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>APIC to Device Management</td>
<td>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.</td>
</tr>
<tr>
<td>Connectivity radio buttons</td>
<td></td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>User Name field</td>
<td>Enter your user name.</td>
</tr>
<tr>
<td>Password field</td>
<td>Enter your password.</td>
</tr>
<tr>
<td>Confirm Password field</td>
<td>Enter your password again.</td>
</tr>
</tbody>
</table>

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

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

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

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>Choose the interface name.</td>
</tr>
<tr>
<td>VNIC</td>
<td>(Only for the virtual device type) Choose a vNIC.</td>
</tr>
</tbody>
</table>
### 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
```
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]

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

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]

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vcenter</td>
<td>(Only for a virtual device) The name of VCenter that hosts the virtual machine for the virtual device.</td>
</tr>
<tr>
<td>vm</td>
<td>(Only for a virtual device) The name of the virtual machine for the virtual device.</td>
</tr>
</tbody>
</table>

**Example:**

For a physical device, enter:

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

For a virtual device, enter:

```plaintext
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]

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vlan</td>
<td>(Only for a physical device) The static encapsulation for the cluster interface. VLAN value must be between 1 to 4094.</td>
</tr>
</tbody>
</table>

**Example:**

For a physical device, enter:

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

For a virtual device, enter:

```plaintext
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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>device</td>
<td>The name of the cluster device that must have been already added to this device cluster using cluster-device command.</td>
</tr>
<tr>
<td>device-interface</td>
<td>The name of the interface on the cluster device.</td>
</tr>
</tbody>
</table>

**Example:**

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

**Step 7**

Add an interface to a member.

```plaintext
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.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>vnic</td>
<td>The name of the vNIC adapter on the virtual machine for the cluster-device. Enclose the name in double quotes.</td>
</tr>
</tbody>
</table>

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:

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

Using an Imported Device with the REST APIs

The following REST API uses an imported device:

```xml
c<polUni>
  <fvTenant dn="uni/tn-tenant1" name="tenant1">
    <vnsLDevIf ldev="uni/tn-mgmt/lDevVip-ADCCluster1"/>
    <vnsLDevCtx ctctxNameOrLbl="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="10.10.10.24"/>
        <vnsLIfCtx connNameOrLbl="outside">
          <vnsRsLIfCtxToLIF tDn="uni/tn-tenant1/lDevIf-[uni/tn-mgmt/lDevVip-ADCCluster1]/lDevIfLIf-outside"/>
          <fvSubnet ip="70.70.70.24"/>
        </vnsLIfCtx>
      </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.

```bash
tenant tenant_name
```

**Example:**

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

Step 3  Import the device.

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>import-from</td>
<td>Name of the tenant from where to import the device.</td>
</tr>
<tr>
<td>device-cluster</td>
<td>Name of the device cluster to import from the specified tenant.</td>
</tr>
</tbody>
</table>

**Example:**

```bash
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:
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.
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.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connector Name</td>
<td>The connector name or label for the logical interface context. The default is Any.</td>
</tr>
<tr>
<td>Logical Interface</td>
<td>The logical interface identifier.</td>
</tr>
<tr>
<td>Bridge Domain</td>
<td>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.</td>
</tr>
<tr>
<td>L3 Network</td>
<td>The Layer 3 context name. For a copy device, do not choose a Layer 3 network.</td>
</tr>
<tr>
<td>L4-L7 Policy based Routing</td>
<td>The policy-based redirect policy to use with logical device context. For a copy device, do not choose a policy-based redirect policy.</td>
</tr>
<tr>
<td>Permit Logging</td>
<td>The permit logging status for the interface context. The default is false.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connector Name</td>
<td>The connector name or label for the logical interface context. The default is Any.</td>
</tr>
<tr>
<td>Cluster Interface</td>
<td>The unique name of the target interface.</td>
</tr>
</tbody>
</table>

---

Cisco APIC Layer 4 to Layer 7 Services Deployment Guide, Release 2.0(1)
<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Bridge Domain</strong></td>
<td>Enter the associated network of the target.</td>
</tr>
<tr>
<td></td>
<td>For Anycast, the bridge domain should be the same as that used for the node.</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>The associated network of the target should be <em>either</em> a bridge domain or an L3 network.</td>
</tr>
<tr>
<td><strong>L3 Network</strong></td>
<td>Enter the associated network of the target.</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>The associated network of the target should be <em>either</em> a bridge domain or an L3 network.</td>
</tr>
<tr>
<td><strong>L3 Destination (VIP)</strong></td>
<td>Indicates whether this logical interface is terminating the L3 traffic in the service chain.</td>
</tr>
<tr>
<td></td>
<td>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.</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>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 (<strong>L4-L7 Policy Based Redirect</strong>)</td>
</tr>
<tr>
<td></td>
<td>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 (<strong>L4-L7 Policy Based Redirect</strong>)</td>
</tr>
<tr>
<td><strong>L4-L7 Policy Based Redirect</strong></td>
<td>Optional. Choose the Policy Based Redirect policy or <strong>Create L4-L7 Policy Based Redirect</strong>.</td>
</tr>
<tr>
<td><strong>Note</strong></td>
<td>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 <strong>L3 Destination (VIP)</strong> box.</td>
</tr>
<tr>
<td><strong>Custom QoS Policy</strong></td>
<td>Optional. Choose a Custom QoS Policy, the default policy, or <strong>Create Custom QoS Policy</strong>.</td>
</tr>
</tbody>
</table>
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.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
</table>
| 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:

```xml
<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:

```xml
<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
- About Service Graph Template Configuration Parameters, on page 38
- Configuring Service Graph Templates Using the GUI, on page 38
- Creating a Service Graph Template Using the REST APIs, on page 38
- Configuring a Service Graph Using the NX-OS-Style CLI, on page 39

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:
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.

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.

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 Service Graph Templates Using the GUI

You can configure the service graph templates using the GUI.

See Using the GUI, on page 193 for the procedure for configuring the service graph templates.

Creating a Service Graph Template Using the REST APIs

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

```xml
<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
```

Cisco APIC Layer 4 to Layer 7 Services Deployment Guide, Release 2.0(1)
Configuring a Service Graph Using the NX-OS-Style CLI

You can configure a service graph 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]
```
**Configuring a Service Graph Using the NX-OS-Style CLI**

### Configuring a Service Graph

#### 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]
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>service</td>
<td>The name of the service node to add.</td>
</tr>
<tr>
<td>device-cluster-tenant</td>
<td>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.</td>
</tr>
<tr>
<td>device-cluster</td>
<td>Name of the device cluster to use for this service node.</td>
</tr>
<tr>
<td>mode</td>
<td>The deployment mode. Possible values are:</td>
</tr>
<tr>
<td></td>
<td>• ADC_ONE_ARM—Specifies one-arm mode.</td>
</tr>
<tr>
<td></td>
<td>• ADC_TWO_ARM—Specifies two-arm mode.</td>
</tr>
<tr>
<td></td>
<td>• FW_ROUTED—Specifies routed (GoTo) mode.</td>
</tr>
<tr>
<td></td>
<td>• FW_TRANS—Specifies transparent (GoThrough) mode.</td>
</tr>
<tr>
<td></td>
<td>• OTHERS—Specifies any other deployment mode.</td>
</tr>
</tbody>
</table>

If the mode is not specified, then a deployment mode is not used.

**Example:**

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

```
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]
```

**Example:**

```
apic1(config-graph)# connector C1 cluster-interface interface_type
```
## Configuring a Service Graph

### Configuring a Service Graph Using the NX-OS-Style CLI

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>connector</td>
<td>The type of the connector in the service graph. Possible values are:</td>
</tr>
<tr>
<td></td>
<td>• provider</td>
</tr>
<tr>
<td></td>
<td>• consumer</td>
</tr>
<tr>
<td>cluster-interface</td>
<td>The type of the device cluster interface. Possible values are:</td>
</tr>
<tr>
<td></td>
<td>• provider</td>
</tr>
<tr>
<td></td>
<td>• consumer</td>
</tr>
</tbody>
</table>

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** 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
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tenant</td>
<td>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.</td>
</tr>
<tr>
<td>name</td>
<td>Name of the bridge domain.</td>
</tr>
</tbody>
</table>

**Example:**

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

**Step 7** (Optional) Configure the direct server return (DSR) virtual IP address (VIP) for the connector.

```
dsrr-vip ip_address
```

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

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>dsr-vip</td>
<td>The virtual IP address of the DSR for the connector.</td>
</tr>
</tbody>
</table>

**Example:**

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

**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
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>connection</td>
<td>The name of the connection.</td>
</tr>
</tbody>
</table>
### Configuring a Service Graph Using the NX-OS-Style CLI

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| terminal  | Connects a service node to the terminal. Specifies the type of the terminal. Possible values are:  
  • provider  
  • consumer |
| service   | The name of the service node to add. service is used only with terminal. service1 and service2 are used only with intra_service. |
| service1  |  |
| service2  |  |
| connector | The type of the connector. Possible values are:  
  • provider  
  • consumer |
| connector1| connector is used only with terminal. connector1 and connector2 are used only with intra_service. |
| connector2|  |
| 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
napic1(config-graph)# connection CON2 terminal provider service N2 connector provider
napic1(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
napic1(config-graph)# connection CON2 intra_service service1 N1 connector1 provider service2 N2 connector2 consumer
napic1(config-graph)# connection CON3 terminal provider service N2 connector provider
napic1(config-graph)# exit
```

#### Step 9

Exit the configuration mode.

#### Example:

```
apic1(config-tenant)# exit
napic1(config)# exit
```
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:
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 52.

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 44. 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":

```xml
<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:

```xml
<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:

```xml
<13extRsPathL3OutAtt 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 `13extRsPathL3OutAtt` 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:

```xml
<13extIfP name="bLeaf-101">
  <13extRsNodeL30utAtt tDn="topology/pod-l/node-101" rtrId="180.0.0.11"/>
  <13extLIfP name="portlf">
    <13extRsPathL30utAtt 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"/>
    </ospfIfP>
  </13extLIfP>
</13extLNodeP>
```

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 `13extRsPathL3OutAtt` 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

```xml
<polUni>
  <fvTenant name="common">
    <fvCtx name="commonctx">
      <fvRsCtxToExtRouteTagPol tnL3extRouteTagPolName="myTagPol"/>
      <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]" instT="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>
          <ospfExtP areaId="100" areaType="regular" areaCtrl="redistribute"/>
        </l3extLNodeP>
      </l3extOut>
      <ospfIfPol name="ospfIfPol" ctrl="mtu-ignore" nwT="bcast" xmitDelay="1" helloIntvl="10" deadIntvl="40" status="created,modified"/>
    </fvCtx>
  </fvTenant>
</polUni>

<polUni>
  <fvTenant name="tenant1">
    <l3extRouteTagPol tag="213" name="myTagPol"/>
    <fvCtx name="tenant1ctx1">
      <fvRsCtxToExtRouteTagPol tnL3extRouteTagPolName="myTagPol"/>
      <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]" instT="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>
          <l3extInstP name="IntInstP">
            <l3extSubnet ip="30.30.30.100/28" scope="import-security"/>
            <l3extSubnet ip="20.20.20.0/24" scope="import-security"/>
          </l3extInstP>
        </l3extLNodeP>
      </l3extOut>
      <ospfIfPol name="ospfIfPol" ctrl="mtu-ignore" nwT="bcast" xmitDelay="1" helloIntvl="10"deadIntvl="40" status="created,modified"/>
    </fvCtx>
  </fvTenant>
</polUni>
```

Cisco APIC Layer 4 to Layer 7 Services Deployment Guide, Release 2.0(1)
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.

The bleaf-101 policy controls where this 13extOut policy is deployed.

You can deploy virtual services with route peering, although the 13extRsPathL3OutAtt validation with the vnsCIf object is not performed. The datapath will work only if the 13extOut 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:
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" 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"></l3extRsNodeL3OutAtt>
      </l3extLNodeP>
    </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:

```
<i3extLIfP name="portIf">
  <i3extRsPathL3OutAtt 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"/>
  </i3extRsPathL3OutAtt>
</i3extLIfP>
```

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 <i3extLNodeP>) and a loopback interface has been configured:

```
<i3extLNodeP name="bLeaf-101">
  <bgpPeerP addr="40.40.40.102/32 "ctrl="send-com"/>
  <i3extRsNodeL3OutAtt tDn="topology/pod-1/node-101" rtrId="180.0.0.8/28">
    <i3extLoopBackIfP addr="50.50.50.100/32"/>
  </i3extRsNodeL3OutAtt>
</i3extLNodeP>
```

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

```xml
<polUni>
  <fvTenant name="tenant11">
    <l3extOut name="StaticInternal" status="created,modified">
      <l3extLNodeP name="bLeaf-201">
        <ipRouteP ip="20.20.20.0/24"/>
        <ipNextHopP nhAddr="30.30.30.102/32"/>
      </ipRouteP>
      <l3extLIfP name="portIf">
        <ipRouteP ip="20.20.20.0/24"/>
        <ipNextHopP nhAddr="30.30.30.102/32"/>
      </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:

```xml
<vnsLDevCtx ctctNameOrLbl="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/inP-IntInstP" status="created,modified"/>
    <vnsRsLIfCtxToLIf tDn="uni/tn-tenant1/lDevVip-Firewall/lIf-internal"/>
  </vnsLIfCtx>
  <vnsLIfCtx connNameOrLbl="external">
    <vnsRsLIfCtxToInstP tDn="uni/tn-tenant1/out-OspfExternal/inP-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:

```xml
<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:

```xml
<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.
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 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 tenantctx1). 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 L3Outs—L3OutInternet and L3OutInternet2—are deployed on two border leaf switches. There is a contract associated between these L3Outs, 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 13extOuts 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 55.

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 56.

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 56.

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 58.

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 58.

### 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 57.

**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 57.

**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.
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:**

```
api1# configure
```

**Step 2** Enter the configure mode for a tenant.

**Example:**

```
api1(config)# tenant 101
```

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

**Example:**

```
api1(config-tenant)# l4l7 graph g1 contract c1
```

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

**Example:**

```
api1(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:**

```
api1(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:
```
apic(config-connector)# l4l7-peer tenant 101 out l101 epg e101 redistribute bgp
apic(config-connector)# exit
```

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

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

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

Example:
```
apic(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:
```
apic(config)# tenant 102
apic(config-tenant)# rtr-cfg bgp1
apic(config-router)# router-id 1.2.3.5
apic(config-router)# exit
```

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

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

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

Example:
```
apic(config-tenant)# external-13 epg e101 l3out l101
apic(config-tenant-l3ext-epg)# vrf member v101
apic(config-tenant-l3ext-epg)# match ip 101.101.1.0/24
apic(config-tenant-l3ext-epg)# exit
apic(config-tenant)# leaf 101
apic(config-leaf)# vrf context tenant 101 vrf v101 l3out l101
apic(config-leaf-vrf)# ip route 101.101.1.0/24 99.1.1.2
apic(config-leaf-vrf)# exit
apic(config-leaf)# interface ethernet 1/10
apic(config-leaf-if)# vrf member tenant 101 vrf v101 l3out l101
apic(config-leaf-if)# vlan-domain member dom101
apic(config-leaf-if)# no switchport
apic(config-leaf-if)# ip address 99.1.1.1/24
apic(config-leaf-if)# exit
apic(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.

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

The following table provides a troubleshooting command that you can run in the `vsh lc` shell.

<table>
<thead>
<tr>
<th>Command</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>show system internal aclqos prefix</code></td>
<td>Examines the IPv4 prefix association rules on a particular leaf switch and the traffic hit counts on the rules.</td>
</tr>
</tbody>
</table>

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: 10.10.10.0/32
    IP broadcast address: 255.255.255.255
    IP primary address route-preference: 1, tag: 0
```

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 Routers 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 0
  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
```
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
```

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
```

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

```
ciscoasa# show running-config
: Saved
:
: Serial Number: 9AGRM5NBEXG
: Hardware:  ASA v, 2048 MB RAM, CPU Xeon 5500 series 2133 MHz
:
ASA Version 9.3(1)
!
hostname ciscoasa
enable password 8Ry2YjIyt7RRXU24 encrypted
name
!
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 1.1.1.1 255.255.255.252
!
<<...>>
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 65
- About Multi-Node Policy-Based Redirect, on page 67
- About Symmetric Policy-Based Redirect, on page 67
- Policy Based Redirect and Hashing Algorithms, on page 68
- Policy-Based Redirect Resilient Hashing, on page 68
- About Bypass Action, on page 70
- About Policy-Based Redirect with L3Out, on page 74
- PBR Support for Service Nodes in Consumer and Provider Bridge Domains, on page 77
- Guidelines and Limitations for Configuring Policy-Based Redirect, on page 78
- Configuring Policy-Based Redirect Using the GUI, on page 83
- Configuring Policy-Based Redirect Using the NX-OS-Style CLI, on page 84
- Verifying a Policy-Based Redirect Configuration Using the NX-OS-Style CLI, on page 87
- About Layer 1/ Layer 2 Policy-Based Redirect, on page 89
- Guidelines and Limitations for Layer 1/ Layer 2 Policy-Based Redirect, on page 90
- Configuring Layer 1/ Layer 2 PBR Using the APIC GUI, on page 90
- Configuring ASA Requirements for Layer 1/ Layer 2 PBR Using CLI, on page 91
- Verifying Layer 1/ Layer 2 PBR Policy On The Leafs Using CLI, on page 92
- Configuring Layer 1/ Layer 2 PBR Using the REST API, on page 93
- Policy Based Redirect and Tracking Service Nodes, on page 94
- About Location-Aware Policy Based Redirect, on page 98
- Policy-Based Redirect and Service Graphs to Redirect All EPG-to-EPG Traffic Within the Same VRF Instance, on page 101

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:
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.

**About Multi-Node Policy-Based Redirect**

Multi-node policy-based redirect enhances PBR by supporting up to five function 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:

```xml
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"/>
</vnsSvcRedirectPol>
```
Policy Based Redirect and Hashing Algorithms

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
```

---

**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 Type (also called, Symmetric) based algorithm was supported in prior releases.

### 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 reshared 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 resharing 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 14: 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 15: 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 16: 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.

About Bypass Action

Prior to Cisco APIC Release 4.1(2), when Threshold Enable is selected when creating a Layer 4 to Layer 7 policy-based redirect, only two options were available: deny action or permit action.
With these two options, in a multi-node policy-based redirect graph, when one node crosses the low threshold, the following action would occur, depending on which of the two options you selected:

- **deny action**: Traffic is dropped at this node.
- **permit action**: Traffic is sent directly to the destination, and the rest of the service chain is skipped.

Beginning with Cisco APIC Release 4.1(2), a new **bypass action** option is available. With this option, in a multi-node policy-based redirect graph, when one node crosses the low threshold, traffic is still able to proceed through the rest of the service chain that is either up or cannot be bypassed.

The following sections describe how traffic is handled for each of these three options using this example two-node policy-based redirect graph.

When both nodes are up, this two-node policy-based redirect behaves in the following manner:

<table>
<thead>
<tr>
<th>Source EPG</th>
<th>Destination EPG</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>300</td>
<td>PBR to n1-external</td>
</tr>
<tr>
<td>201</td>
<td>300</td>
<td>PBR to n2-external</td>
</tr>
<tr>
<td>302</td>
<td>300</td>
<td>permit</td>
</tr>
<tr>
<td>300</td>
<td>100</td>
<td>PBR to n2-internal</td>
</tr>
<tr>
<td>202</td>
<td>100</td>
<td>PBR to n1-internal</td>
</tr>
<tr>
<td>101</td>
<td>100</td>
<td>permit</td>
</tr>
</tbody>
</table>

The following sections describe how the two-node policy-based redirect behaves when the first node goes down, based on the option that you select in the **Threshold Down Action** field.

**deny action**

Using the example configuration described above, if you select **deny action** in the **Threshold Down Action** field and the first node goes down, the PBR policies that use the first node are updated to "Drop", and communication between the Client EPG and the Web EPG will be dropped, as shown in the following table.

<table>
<thead>
<tr>
<th>Source EPG</th>
<th>Destination EPG</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>300</td>
<td>Drop</td>
</tr>
</tbody>
</table>
Using the example configuration described above, if you select **permit action** in the **Threshold Down Action** field and the first node goes down, the PBR policies that use the first node are updated to "Permit". Traffic from the Client EPG to the Web EPG (from 100 to 300) proceeds directly, without the service node. Return traffic from the Web EPG to the Client EPG (from 300 to 100) is redirected to n2-internal, as shown in the following table; however, the second node might drop the packet because it is an asymmetric flow.

<table>
<thead>
<tr>
<th>Source EPG</th>
<th>Destination EPG</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>300</td>
<td>Permit</td>
</tr>
<tr>
<td>201</td>
<td>300</td>
<td>PBR to n2-external</td>
</tr>
<tr>
<td>302</td>
<td>300</td>
<td>permit</td>
</tr>
<tr>
<td>300</td>
<td>100</td>
<td>PBR to n2-internal</td>
</tr>
<tr>
<td>202</td>
<td>100</td>
<td>Permit</td>
</tr>
<tr>
<td>101</td>
<td>100</td>
<td>permit</td>
</tr>
</tbody>
</table>

**bypass action**

Beginning with Cisco APIC Release 4.1(2), if you select the new **bypass action** option in the **Threshold Down Action** field and the first node goes down, the PBR policies that use the first node are updated to "PBR to next device". In this case, the following occurs:

- Traffic from the Client EPG to the Web EPG (from 100 to 300) is redirected to n2-external.
- Return traffic from the Web EPG to the Client EPG (from 300 to 100) is redirected to n2-internal.
- Return traffic from n2-external to consumer is set to "Permit".

<table>
<thead>
<tr>
<th>Source EPG</th>
<th>Destination EPG</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>300</td>
<td>PBR to n2-external</td>
</tr>
<tr>
<td>201</td>
<td>300</td>
<td>PBR to n2-external</td>
</tr>
<tr>
<td>302</td>
<td>300</td>
<td>permit</td>
</tr>
<tr>
<td>300</td>
<td>100</td>
<td>PBR to n2-internal</td>
</tr>
</tbody>
</table>
### Guidelines and Limitations

Following are the guidelines and limitations for the **bypass action** option:

- The **bypass action** option is supported only on new generation ToR switches, which are switch models with "EX", "FX" or "FX2" at the end of the switch name.
- The **bypass action** option is not needed on a one-node service graph. If bypass is configured in such a case, forwarding behavior is the same as permit action.
- L3Out EPGs and regular EPGs can be consumer or provider EPGs.
- A service node that has NAT enabled cannot be bypassed, as that will break the traffic flow.
- The **bypass action** option is not supported in the following cases:
  - Layer 4 to Layer 7 devices in one-arm mode
  - Layer 1/Layer 2 PBR nodes
  - Remote leaf switches
- Do not use the same PBR policy in more than one service graph if bypass action is enabled. Cisco APIC will reject configurations if the same PBR policy with bypass action is used in multiple service graphs. To avoid this, configure different PBR policies that use the same PBR destination IP address, MAC address and Health Group.

### Configuring the Threshold Down Action in Layer 4 to Layer 7 Policy-Based Redirect

#### Before you begin

This task assumes that an Layer 4 to Layer 7 Policy-Based Redirect policy has been created.

<table>
<thead>
<tr>
<th>Source EPG</th>
<th>Destination EPG</th>
<th>Action</th>
</tr>
</thead>
<tbody>
<tr>
<td>202</td>
<td>100</td>
<td>Permit</td>
</tr>
<tr>
<td>101</td>
<td>100</td>
<td>permit</td>
</tr>
</tbody>
</table>

#### 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 **Destination Type** field, select **L3**.

#### Step 5
In the **IP SLA Monitoring Policy** field, select an existing policy or create a new IP SLA monitoring policy.

For more information on creating a new IP SLA monitoring policy, see Configuring IP SLA Monitoring Policy Using the GUI, on page 97.

#### Step 6
Check the **Threshold Enable** check box.
The following fields appear:

- Min Threshold Percent (percentage)
- Max Threshold Percent (percentage)
- Threshold Down Action

**Step 7**
Select the minimum and maximum threshold percentage.
For more information on the minimum and maximum thresholds, see Threshold Settings, on page 95.

**Step 8**
In the **Threshold Down Action** area, select the threshold down action.
The options are:
- bypass action
- deny action
- permit action

**Step 9**
Click Submit.

---

**About Policy-Based Redirect with L3Out**

Prior to Cisco APIC Release 4.1(2), if policy-based redirect is enabled on a node, both the consumer and the provider connectors of the node must be in bridge domains, as shown in the following illustration, even though the policy-based redirect is not required on an interface.

![Diagram showing policy-based redirect with L3Out](image)

Beginning with Cisco APIC Release 4.1(2), uni-directional policy-based redirect with L3Out is now supported, as shown in the following illustration. In this example, policy-based redirect is enabled in the bridge domain in the consumer connector, but the policy-based redirect is not enabled on the provider connector in the L3Out.
Configuring Policy-Based Redirect with L3Out

The configuration steps for policy-based redirect with L3Out is essentially the same as a typical policy-based redirect configuration. It is assumed that you already have the necessary tenant, VRF, EPGs, and bridge domains for the EPGs and service bridge domains configured.

1. Create a Layer 4 to Layer 7 device. For the concrete interface, the path should match with the VLAN used in the L3Out logical interface.

2. Create a Service Graph template.

3. Create a PBR policy.


5. Create a Device Selection Policy using the following settings:
   - Enable the PBR on the consumer connector and select the L3Out on the provider connector.
   - Deselect the L3 Destination (VIP) field (remove the check from the checkbox) on both the consumer and provider connector for the node with L3Out.

   For example, you would make the following selections in the Logical Interface Connector page for each connector:
   - On the consumer side, in the Associated Network field, select Bridge Domain and enable PBR in the L4-L7 Policy-Based Redirect field. Also deselect the L3 Destination (VIP) field.
   - On the provider side, in the Associated Network field, select L3 External Network and select the L3Out in the L3 External Network field. Also deselect the L3 Destination (VIP) field.

6. Apply the Service Graph, where you attach the Service Graph to the contract.

Note that you can configure QoS as part of the L3Out configuration.

Example Use Case for Policy-Based Redirect with L3Out

An example situation where policy-based redirect with L3Out would be useful would be a configuration with symmetric PBR with multiple network address translation (NAT) devices, as shown in the following figure.
In this situation, for internal to external traffic, symmetric PBR is used to load balance traffic to one of the PBR nodes. In addition, the IP address from the NAT-pool is outside of the service bridge domain subnet range. The return traffic will go back to the same node in this situation, but for releases prior to Cisco APIC Release 4.1(2), the L3Out cannot be used on the NAT-external side because if the PBR is used on the service node, the other service node connector must be in the bridge domain as well.

However, beginning with Cisco APIC Release 4.1(2), support is available for enabling PBR on the consumer connector, with the provider connector as an L3Out, as shown in the following figure.
**Guidelines and Limitations**

Following are the guidelines and limitations for policy-based redirect with L3Out:

- Use a specific L3Out EPG subnet if there are other L3Out EPGs in the same VRF; otherwise, the other L3Outs might be used by mistake.

- Ensure that IP translation occurs on the service node. If the SNAT is not properly done on the firewall, it could be classified to the L3Out internal and could cause a loop.

- L3Out is supported only on the provider side of the last node.

---

**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.
Guidelines and Limitations for Configuring Policy-Based Redirect

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

• 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.

• The policy-based redirect bridge domain must have the endpoint dataplane learning disabled.

• Provision service appliances in a separate bridge domain. Starting with the Cisco Application Policy Infrastructure Controller (Cisco APIC) release 3.1(x), 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-C93180Y-EX and N9K-93108TC-EX switches, the service appliance can be in the same leaf switch as either the source or destination endpoint group.

• The service appliance can only be in a regular 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.

• Multi-node policy-based redirect (multi-node PBR):
  • Supports up to five function nodes in a service graph that can be configured for policy-based redirect.

  • 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.
• Redirection to transparent services 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 17: Supported Policy-based Redirect Configurations in the Same VRF Instance

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

- VRF1
- VRF2
- EPG Client
- EPG Web
- BD1 (192.168.1.254/24)
- BD2 (192.168.2.254/24)
- Svc-BD1
- Svc-BD2
- Unicast Routing: Yes

**Figure 18: Supported Policy-based Redirect Configurations in a Different VRF Instance**
Figure 19: 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:

1. Select the Service Type as Other.
2. Select the Device Type Physical (cloud/virtual is not supported).
3. Select a physical domain.
4. Select the Function Type L1 or L2 as required.
5. Create external and internal concrete interfaces and port connectivity on the corresponding leafs.
6. 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 nodes.

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.
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:**

```
1417 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
```

**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
    1417 graph PBRv6_ASA_HA_Mode_Graph
  exit
  exit
vrf context CTX1
```
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:
1417 cluster import-from T1 device-cluster ifav-asa-vm-ha

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

Example:
1417 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
Step 2

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

Example:

```
apic1# show running-config tenant PBRv6_ASA_HA_Mode 1417 graph PBRv6_ASA_HA_Mode_Graph
# Command: show running-config tenant PBRv6_ASA_HA_Mode 1417 graph PBRv6_ASA_HA_Mode_Graph
tenant PBRv6_ASA_HA_Mode
  1417 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

  connection C1 terminal consumer service N2 connector consumer
  connection C2 terminal provider service N2 connector provider
  exit
```

Step 3

Show the service graph configuration.

Example:

```
apic1# show 1417-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 Mode Name : N2
Connector Encap Bridge-Domain Device Interface Service Redirect Policy
----------------- ------------------ ------------------ ----------------------
consumer vlan-241 PBRv6_ASA_HA_Mode consumer_PBRv6 External_leg
  External-BD3
provider vlan-105 PBRv6_ASA_HA_Mode provider_PBRv6 Internal_leg
  Internal-BD4
```
About Layer 1/ Layer 2 Policy-Based Redirect

Prior to ACI release 4.1, you could only deploy a L4-L7 device operating in Layer 1 or Layer 2 mode by using Service Graph and defining the L4-L7 device in Go-Through mode.

Starting with ACI release 4.1, you can deploy a L4-L7 device operating in Layer 1 / Layer 2 mode with Service Graph PBR also.

Starting from APIC release 4.1, PBR can be configured to redirect traffic to a L4-L7 device configured in Layer 1/ Layer 2 device mode as well.

As part of L1/L2 PBR feature, APIC can verify whether the L4-L7 device is forwarding traffic by using L2 Ping packets for link layer tracking.

The advantages compared to configuring a Service Graph with a L4-L7 device in Go-Through mode with Layer 1 / Layer 2 PBR are that you don’t need to create two bridge domains to insert a L4-L7 device between endpoints in the same subnet, hence the configuration is simplified and there are less risks to introduce loops by bridging Bridge Domains.

Differently from Go-Through mode, which can forward also non-IP traffic, Layer 1 / Layer 2 PBR can only forward IP-packets.

The following list summarizes some of the key Layer 1 / Layer 2 PBR configuration concepts:

• Deploying a L4-L7 device with Layer 1/ Layer 2 PBR requires the configuration of two service bridge domains, one for the consumer-side and one for the provider-side, unlike regular PBR, these bridge domains cannot be the same as the bridge domains where the endpoints (clients or servers) are configured.

• The service bridge domains must be configured for routing.

• The physical L4-L7 device can be connected with individual links or with vPC to the leaf switches.

• With a Layer 1 device, the consumer side VLAN and the provider side VLAN are the same but on different bridge domains, hence the consumer and provider-side of the L4-L7 device must be on different physical leafs.

• When the L4-L7 device is configured as a Layer 1 or Layer 2 device, it doesn’t have an IP address on the interface where it receives and sends traffic, hence the redirect policy is defined by entering the leaf/port and VLAN where it is connected to.

• The redirect policy configuration requires only the definition of the leaf/port and VLAN, entering a MAC address is optional. If the MAC field is left empty, ACI generates dynamically one MAC address that is used to rewrite the destination MAC address when sending to the L4-L7 device on the service bridge domain. These MAC addresses are not the L4-L7 device MAC addresses. They are virtual MAC addresses that ACI uses to rewrite the destination MAC address of the traffic.

• If the L4-L7 device is deployed in Layer 2 mode, it must be configured statically to forward the MAC addresses that PBR uses to forward traffic to the L4-L7 device. One MAC address identifies the consumer-to-consumer destination MAC address used on the service bridge domains and the other MAC address defines the provider-to-consumer destination MAC address used on the service bridge domains. These MAC addresses can be entered manually by the user in APIC as part of the redirect policy definition, or they are auto-generated if the field is left empty. The admin has to enter these MAC addresses in the configuration of the L4-L7 device in the MAC address table and be associated with the provider-side port for the MAC address used in the consumer-to-provider direction and with the consumer-side port for the provider-to-consumer direction.
• Layer 1/ Layer 2 PBR is based on forwarding to a leaf/port/VLAN, hence it can only be deployed with physical domains not with VMM domains. If you need to deploy Layer 1/ Layer 2 PBR with a virtual appliance, that must be configured with a physical domain.

• From a high availability perspective, the L4-L7 device is deployed in Active/Standby mode and ACI has to perform tracking in order to verify which path (leaf/port) is active and which one is standby. With Layer 1/ Layer 2 PBR, the admin has to configure tracking as part of the HA configuration.

• Layer 1 and Layer 2 devices use L2 ping for link layer tracking as the L4-L7 device does not respond to TCP or ICMP ping. IP SLA type should be l2ping.

Guidelines and Limitations for Layer 1/ Layer 2 Policy-Based Redirect

Observe the following guidelines and limitations when planning Layer 1/ Layer 2 Policy-Based Redirect service nodes:

• Layer 1 / Layer 2 Policy-Based Redirect is not supported from CLI.

• Active-active deployment/ECMP paths are not supported for Layer 1/ Layer 2 PBR devices.

• The two legs of the Layer 1 service device need to be configured on a different leaf switch to avoid packet loops. Per port VLAN is not supported.

• Shared bridge domain is not supported. A Layer 1/ Layer 2 device bridge domain cannot be shared with Layer 3 device or regular EPGs.

• Service node in managed mode is not supported.

• Layer 1/ Layer 2 devices support physical domain only, VMM domain is not supported.

• As active-active is not supported, threshold is not applicable. Down action is deny when tracking is enabled. Down action permit can not be set.

• Tracking is mandatory when service devices are in Active/Standby HA mode.

• For both Layer 1 and Layer 2 PBR, MAC is an optional parameter. APIC assigns the MAC address if it is not configured. In both cases, traffic is routed to the device using the MAC address which can be user configured or implicitly assigned by APIC.

Configuring Layer 1/ Layer 2 PBR Using the APIC GUI

Before you begin

• Create a L4-L7 device and service graph using the Layer 1/ Layer 2 function type, see configuration steps in Configuring Policy-Based Redirect Using the GUI.

Step 1  On the menu bar, choose Tenants > All Tenants.
Step 2  In the Navigation pane, choose Tenant tenant_name > Policies > Protocol > L4-L7 Policy Based Redirect.
Step 3 In the Work pane, choose Action > Create L4-L7 Policy Based Redirect.

Step 4 In the Create L4-L7 Policy Based Redirect dialog box, complete the following fields:
   a) In the Name field, provide a name.
   b) In the Destination Type field, select L1 or L2.
   c) Expand the L1/L2 Destinations table and complete the required fields to create the redirected destination.
   d) Click OK.

**Note** Do not enter an actual interface MAC address of ASA. Either leave it blank so the APIC generates MAC automatically or enter a dummy MAC address for external policy MAC A and internal PBR policy MAC B. Remember these MAC addresses are used in ASA configuration.

Step 5 Click Submit.

Step 6 In the Navigation pane, choose Services > L4-L7 > Device Selection Policies > Logical Device Context_name.

Step 7 Expand the Logical Device and apply the Layer 1/ Layer 2 PBR policy in the L4-L7 Policy-Based Redirect field for the consumer or provider.

Step 8 Click Submit.

---

## Configuring ASA Requirements for Layer 1/ Layer 2 PBR Using CLI

**Step 1** The ASA interfaces (service legs) need to be configured in the same bridge-group.

**Example:**
```
interface GigabitEthernet0/0
nameif externalIf
bridge-group 1

interface GigabitEthernet0/1
nameif internalIf
bridge-group 1
```

The ASA does not have a flood behavior. It drops any packet with an unknown destination MAC address that it has not learned. The traffic and L2 ping packet does not use the service EP MAC address as the source MAC. It is always used as destination MAC. The ASA learns from the source MAC. MAC learning is disabled to avoid conflicting entries getting created on the ASA as L2 ping uses the same source MAC to track external and internal.

**Note** Apart from the configuration inside the ASA, if the ASA is running as a VM in vcenter, the vswitch requires to be configured in promiscuous mode to allow L2 ping. For Layer 2, port-groups in vswitch need to be in different VLANs, while for Layer 1, they need to be in same VLAN.

**Step 2** In the following example, `externalIf` is the interface name on ASA, which is used as a consumer connector of the Layer 1 / Layer 2 service node. `internalIf` is the interface name on ASA, which is used as a provider connector of the Layer 1 / Layer 2 service node. Disable MAC learning on `externalIf` and `internalIf`. L2 ping uses the same source MAC when it tries to track both the external and internal legs.

**Example:**
Step 3
Configure ASA rules to permit L2 ping custom ethertype.

Example:
access-list Permit-Eth ethertype permit any
access-group Permit-Eth in interface externalIf
access-group Permit-Eth in interface internalIf

Step 4
Static MAC address table entries need to be added for L2 ping and PBR traffic to be forwarded to the appropriate interface. For example, if the service EP for consumer to provider traffic redirection has MAC A and the service EP for provider to consumer traffic redirection has MAC B:

Example:
mac-address-table static externalIf (MAC B)
mac-address-table static internalIf (MAC A)

Verifying Layer 1/ Layer 2 PBR Policy On The Leafs Using CLI

The example commands in this procedure are for configuring Layer 1 and Layer 2 Policy-Based Redirect nodes.

Step 1
Check whether PBR group and destination information are configured on the switch:

Example:
sdk74-leaf4# show service redir info
GrpID Name destination operSt
------ ----- ---------------- ---
1 destgrp-1 dest-[50.50.50.1]-[vxlan-2719744] enabled
2 destgrp-2 dest-[20.20.20.1]-[vxlan-2719744] enabled
Name bdVnil ip vMac vrf vrfEncap operSt
--- ------- --- --- --- --- --- ---
dest-[20.20.20.1]-[vxlan-2719744] vxlan-16514958 20.20.20.1 00:00:14:00:00:01 coke1:cokectx1 enabled
dest-[50.50.50.1]-[vxlan-2719744] vxlan-16711542 50.50.50.1 00:00:3C:00:00:01 coke1:cokectx1 enabled

Step 2
Check whether zoning-rule is configured with correct action and group information:

Example:
sdk74-leaf4# show zoning-rule | grep redir
4103 49155 49154 18 enabled 2719744
dir(destgrp-2) fully_qual(6)
4106 49154 49155 17 enabled 2719744
dir(destgrp-1) fully_qual(6)

Step 3
Aclqos subcommand for PBR:

Example:
module-1# sh system internal aclqos services redir ?
<CR>
Step 4  
**Zoning-rule command:**

**Example:**

```bash
module-1# sh system internal aclqos zoning-rules 4106
ASIC type is Sug

-----------------------------------------------
Rule ID: 4106 Scope 3 Src EPG: 49154 Dst EPG: 49155 Filter 17
Redir group: 1

Curr TCAM resource:

-----------------------------------------------
  unit_id: 0
  --- Region priority: 1539 (rule prio: 6 entry: 3)---
    sw_index = 44 | hw_index = 44
  --- SDK Info ---
    Result/Stats Idx: 81876
    Tcam Total Entries: 1
    HW Stats: 0
```

---

**Configuring Layer 1/ Layer 2 PBR Using the REST API**

Layer 1/ Layer 2 Policy-Based Redirect configuration:

**Example:**
<polUni>
  <fvTenant name="coke" >
    <!-- For L2 device -->
    <vnsLDevVip name="N1" funcType="L2" managed="no"/>
  </fvTenant>
  <fvTenant name="coke" >
    <!-- For L1 device -->
    <vnsLDevVip name="N1" funcType="L1" managed="no"/>
  </fvTenant>

  <vnsAbsGraph descr="" dn="uni/tn-coke/AbsGraph-WebGraph" name="WebGraph" ownerKey="" ownerTag=""
    uiTemplateType="UNSPECIFIED">
    <!-- For L2 device -->
    <vnsAbsNode descr="" funcTemplateType="OTHER" funcType="L2" isCopy="no" managed="no" name="N1"
      ownerKey="" ownerTag="" routingMode="Redirect" sequenceNumber="0" shareEncap="no"/>
  </vnsAbsGraph>

  <fvIPSLAMonitoringPol name="Pol2" slaType="l2ping"/>
  <vnsSvcCont>
    <vnsRedirectHealthGroup name="2" />
    <vnsSvcRedirectPol name="N1Ext" destType="L2">
      <vnsRsIPSLAMonitoringPol tDn="uni/tn-coke/ipslaMonitoringPol-Pol2"/>
      <vnsL1L2RedirectDest destName="1"/>
    </vnsSvcRedirectPol>
  </vnsSvcCont>

  <fvTenant name="coke" >
  </fvTenant>
</polUni>

Policy Based Redirect and Tracking Service Nodes

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).

With Cisco APIC, Release 2.2(3x), Policy Based Redirect feature (PBR) supports tracking service nodes:
Redirect Destination nodes support dual IP stack. Therefore, both IPv4 and IPv6 addresses can be configured at the same time.

Switches internally use the Cisco IP SLA monitoring feature to support the PBR tracking. The tracking feature marks a Redirect Destination node down if the service node is not reachable. The tracking feature marks a Redirect Destination node up if the service node resumes connectivity. When a service node is marked 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 down. Hence, both ingress and egress traffic gets hashed to a different service node in the cluster of the Redirect Destination nodes.

**Threshold Settings**

The following threshold settings are available when configuring the PBR policy for tracking service nodes:

- **Threshold enabled or disabled:** When the threshold is enabled, you can 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 so 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 an operational state, a percentage greater than the maximum threshold percentage must be reached first.

**Guidelines and Limitations for Policy Based Redirect and Tracking Service Nodes**

Follow these guidelines and limitations when utilizing PBR tracking and service nodes:

- **Destination groups that share destinations must have same health group and IP SLA monitoring policies configured.**

- **Starting in release 4.0(1), remote leaf configurations support PBR tracking, but only if system-level Global GIPo is enabled. See **Configuring Global GIPo for Remote Leaf Using the GUI**.**

- **Starting in release 4.0(1), remote leaf configurations support PBR resilient hashing.**

- **Multipod fabric setup is supported. Multi-site setup is not supported.**
• Layer 3 Out is supported for consumer and provider EPGs.
• TCP or ICMP protocol types are used to track the Redirect Destination nodes.
• Maximum number of trackable IP addresses supported by Policy Based Redirect is 100 in leaf switches and 200 in the ACI fabric.
• Maximum number graph instances in the ACI fabric is 1000 per fabric.
• Maximum number of graph instances per device is 100.
• Maximum number of service nodes that can be configured for PBR is 40 per policy.
• Maximum number of service nodes supported in one Service Chain is three.
• Shared services are supported with PBR tracking.
• The following threshold down actions are supported:
  • bypass action
  • deny action
  • permit action

### 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.
You are here

1. 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 click 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 L4-L7 Redirect Health Policy is configured.

### Configuring IP SLA Monitoring Policy Using the GUI

To enable Cisco APIC to send monitoring probes for a specific SLA type using the APIC GUI, perform the following steps:

**Step 1**
On the menu bar, click Tenant > tenant_name. In the navigation pane, click Policies > Protocol > IP SLA.

**Step 2**
Right-click IP SLA Monitoring Policies, and click Create IP SLA Monitoring Policy.

**Step 3**
In the Create IP SLA Monitoring Policy dialog box, perform the following actions:

a) In the Name field, enter a unique name for the IP SLA Monitoring policy.

b) In the SLA Frequency field, enter the interval probe time in seconds. The minimum interval time is 1 second.

   **Note** The failure detection time is three times the probe interval.

   c) In the SLA Type field, choose the SLA type. Click Submit.

      The SLA type can be TCP or ICMP. ICMP is the default value.

      **Note** L2Ping is supported only for L1/L2 PBR tracking.

   d) If you chose TCP, enter a port number in the Destination Port field.

The IP SLA Monitoring Policy is configured.

### Configuring Global GIPO for Remote Leaf Using the GUI

Performing this task allows PBR tracking to function in remote leaf configurations.
This configuration must be performed for PBR tracking to function on a remote leaf. Without this configuration, PBR tracking will not work on the remote leaf, even when the main data center is reachable.

Note

Step 1
On the menu bar, click **System > System Settings**.

Step 2
In the **System Settings** navigation pane, click **System Global GIPo**.

Step 3
In the **System Global GIPo Policy** work pane, click **Enabled**.

Step 4
In the **Policy Usage Warning** dialog, review the nodes and policies that may be using the GIPo policy and, if appropriate, click **Submit Changes**.

### Configuring PBR to Support Tracking Service Nodes Using the REST API

Configure PBR to support tracking service nodes.

Example:

```xml
<polUni>
  <fvTenant name="coke">
    <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-coke/svcCont/redirectHealthGroup-fwService1"/>
        </vnsRedirectDest>
        <vnsRsIPSLAMonitoringPol tDn="uni/tn-coke/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-coke/svcCont/redirectHealthGroup-fwService1"/>
        </vnsRedirectDest>
        <vnsRsIPSLAMonitoringPol tDn="uni/tn-coke/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 example 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

<table>
<thead>
<tr>
<th>PBR node</th>
<th>IP</th>
<th>MAC</th>
<th>Pod #</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>192.168.3.1</td>
<td>A</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>192.168.3.2</td>
<td>B</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>192.168.3.3</td>
<td>C</td>
<td>2</td>
</tr>
<tr>
<td>D</td>
<td>192.168.3.4</td>
<td>D</td>
<td>2</td>
</tr>
</tbody>
</table>

Guidelines for Location-Aware PBR

Follow these guidelines when utilizing 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.
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" podId="2">
      <vnsRedirectDest ip="40.40.40.100" mac="00:00:00:00:00:01" podId="2"/>
    </vnsSvcRedirectPol>
  </vnsSvcCont>
</fvTenant>
```
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:
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
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.
- 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 an 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 **Creating a Layer 4 to Layer 7 Device Using the GUI, on page 15**.

**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**.
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:
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*

![Diagram showing 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 a direct server return (DSR) static service deployment:

```xml
<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"/>
  </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">
  </fvAEPg>
</fvAp>
```
The DSR configuration is downloaded to all top-of-rack switches (ToRs) where the EPG with a Layer 4 to Layer 7 virtual IP address is deployed or an EPG that has a contract with the EPG with a Layer 4 to Layer 7 virtual IP address is deployed, regardless of the contract direction. In the example, the DSR virtual IP address configuration is downloaded to the ToR nodes 101, 103, and 104. Node 104 has a load balancer EPG with a Layer 4 to Layer 7 virtual IP address configured. Nodes 101 and 103 have a webserver or client EPG, which has a contract to the load balancer EPG.

All ToRs that downloaded the DSR configuration do not learn the Layer 4 to Layer 7 virtual IP address from the datapath. Such ToRs also do not learn the Layer 4 to Layer 7 virtual IP address from the other EPGs. This is true even if you use Address Resolution Protocol (ARP), Gratuitous Address Resolution Protocol (GARP), or IPv6 Neighbor Discovery (ND). For example, the ToRs only learn the Layer 4 to Layer 7 virtual IP address from a load balancer EPG by way of the control plane. This restriction helps to prevent the erroneous learning of the Layer 4 to Layer 7 virtual IP address from a web server EPG, such as if you forgot to suppress ARP on a web server.

**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](image)

**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:
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 SvcVips 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:

```plaintext
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:

```bash
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:

Configuring the Device and Chassis Manager

- About Device Managers and Chassis Managers, on page 115
- Device Manager and Chassis Manager Behavior, on page 118
- Creating Devices Using the GUI, on page 119
- Creating a Chassis Using the GUI, on page 119
- Example XML for Device Managers and Chassis Managers, on page 120
- Device and Chassis Callouts, on page 122

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:
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 AL Dev 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 MDevS, 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:
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:

```xml
<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:

```xml
<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:

```xml
<polUni>
  <fvTenant name="tenant1">
    <vnsLDevVip name="InsiemeCluster" devtype="VIRTUAL">
      ...
      <vnsRsMDevAtt tDn="uni/infra/mDev-Insieme-Generic-1.0"/>
      <vnsRsLDevToDevMgr 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:

```xml
<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:

```xml
<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:

```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'},
                          ...}}},
  '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 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': {'10.10.10.11': {'port': 1234},
                     '10.10.10.12': {'port': 1234},
                     '10.10.10.13': {'port': 1234},
               'name': 'Foo'},
               'port': 80,
               'version': '1.0',
               'virtual': True}}
```
Example 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 14071921972992 [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', 'password': '<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}),
 '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': 'admin', '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 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': {'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', 'password': '<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', 'password': '<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

```
{"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 127
- About Managed and Unmanaged Logical Devices, on page 127
- About Managed and Unmanaged Function Nodes, on page 128
- About Layer 4 to Layer 7 Services Endpoint Groups, on page 129
- Using Static Encapsulation for a Graph Connector, on page 129
- Creating a Physical Device Using the NX-OS-Style CLI, on page 129
- Creating a High Availability Cluster Using the NX-OS-Style CLI, on page 131
- Creating a Virtual Device Using the NX-OS-Style CLI, on page 132
- Example XML for the Unmanaged Mode, on page 133
- Unmanaged Mode Behavior, on page 135

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:

```xml
<!-- Specified if the device is a managed device-->
<property name="managed"/>
```
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 (vnsLDevVlP) 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:

```xml
<!-- 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 134.

### 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 134.

### 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 Availability 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
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.

Example:
tenant tenant_name

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

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:

```xml
<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:

```xml
<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:

```xml
<fvTenant name="HA_Tenant1">
  <vnsAbsGraph name="g1">
    <vnsAbsTermNodeProv name="Input1">
      <vnsAbsTermConn name="C1"/>
    </vnsAbsTermConn>
  </vnsAbsTermNodeProv>
</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:

```xml
<vnsAbsGraph>
  <vnsAbsNode name="Node1" managed="no">
    <vnsAbsFuncConn name="outside" />
    <vnsAbsFuncConn name="inside" />
  </vnsAbsNode>

  <vnsAbsTermNodeCon name="Output1">
    <vnsAbsTermConn name="C6" />
  </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 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 service endpoint group:
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.
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.

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 139.

Step 2
Create a service graph template to use for copy services.

For information about creating a service graph template, see Creating a Layer 4 to Layer 7 Service Graph Template Using the GUI, on page 196.

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 198.

---

### 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 138.

**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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name field</td>
<td>Enter a name for the copy device.</td>
</tr>
<tr>
<td>Device Type buttons</td>
<td>The device type. A copy device can only be a physical device.</td>
</tr>
<tr>
<td>Physical Domain drop-down list</td>
<td>Choose the physical domain for the device.</td>
</tr>
</tbody>
</table>

**Step 6**
In the Device 1 section, click + to add a device interface, complete the following fields, and then click Update:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name field</td>
<td>Enter a name for the device interface.</td>
</tr>
<tr>
<td>Path drop-down list</td>
<td>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.</td>
</tr>
</tbody>
</table>

**Step 7**
In the Cluster section, click + to add a cluster interface, complete the following fields, and then click Update:
### 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:**

```
1417 cluster name Copy_1 type physical vlan-domain phys_scale_copy service COPY function none
custom-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:**

```
1417 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 copy-service CP1 connector copy
Exit
```
Step 3  Attach the contract to the graph.

**Example:**

```
contract c5
  scope tenant
  subject Subject
    access-group default both
  1417 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
  1417 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 copy service CP2
  connection C2 terminal provider service Fire connector provider copy service CP3
  exit
  Exit
```

The following example creates a firewall and load balance in one-arm mode with copy devices attached in all the links:

```
1417 graph Graph_LB_Firewall contract c1_firewall
  service Fire device-cluster-tenant Tenant_Firewall_LB device-cluster Firewall_1 mode
```
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:**
```xml
<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 name="copy_Dyn_Device_0" vcenterName="" vmName="">
<vnsRsCIfPathAtt tDn="topology/pod-1/paths-104/pathep-[eth1/15]"/>
<vnsCIf>
<vnsRsCIfAttN tDn="uni/tn-t22/lDevVip-copy0/cDev-copy_Dyn_Device_0/cIf-[int1]"/>
<vnsRsCIfAttN tDn="uni/tn-t22/lDevVip-copy0/cDev-copy_Dyn_Device_0/cIf-[int2]"/>
</vnsCIf>
</vnsCDev>
<vnsLIf encap="vlan-3540" name="TAP">
<vnsRsCIfAttN tDn="uni/tn-t22/lDevVip-copy0/cIf-[int1]"/>
<vnsRsCIfAttN tDn="uni/tn-t22/lDevVip-copy0/cIf-[int2]"/>
</vnsLIf>
</vnsLDevVip>
```

**Step 2** Create a logical device context (also known as a device selection policy).

**Example:**
```xml
<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-copy0/1If-TAP"/>
</vnsLIfCtx>
</vnsLDevCtx>
```

**Step 3** Create and apply the copy graph template.

**Example:**
```xml
<vnsAbsGraph descr="" name="g0" ownerKey="" ownerTag="" uiTemplateType="UNSPECIFIED">
<vnsAbsTermNodeCon descr="" name="T1" ownerKey="" ownerTag="" />
<vnsInTerm descr="" name="" ownerKey="" ownerTag="" />
<vnsOutTerm descr="" name="" />
</vnsAbsTermNodeCon>
<vnsAbsTermNodeProv descr="" name="T2" ownerKey="" ownerTag="" />
<vnsAbsTermConn descr="" name="1" ownerKey="" ownerTag="" />
<vnsInTerm descr="" name="" ownerKey="" ownerTag="" />
<vnsOutTerm descr="" name="" />
</vnsAbsTermNodeProv>
<vnsAbsConnection adjType="L2" connDir="provider" connType="external" descr="" name="C1" ownerKey="" ownerTag="" unicastRoute="yes">
<vnsRsAbsConnectionConn tDn="uni/tn-t22/AbsGraph-g0/AbsTermNodeCon-T1/AbsTConn"/>
<vnsRsAbsConnectionConn 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="" routingNode="unspecified" sequenceNumber="0" shareEncap="no">
<vnsAbsFuncConn 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:

```xml
<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:

```xml
<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

Configuration Parameters

• Configuration Parameters Inside the Device Package Specification, on page 145
• Configuration Parameters Inside An Abstract Function Profile, on page 148
• Configuration Parameters Inside an Abstract Function Node in a Service Graph, on page 152
• Configuration Parameters Inside Various Configuration MOs, on page 155
• Parameter Resolution, on page 158
• Looking Up an MO During Parameter Resolution, on page 159
• About Role-Based Access Control Rule Enhancements, on page 160

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.
MFOLDER

MFOLDER is a group of configuration items that can contain MParams and other nested MFolders. An MFOLDER has following attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>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.</td>
</tr>
<tr>
<td>Description</td>
<td>Describes the configuration item.</td>
</tr>
<tr>
<td>Cardinality</td>
<td>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.</td>
</tr>
<tr>
<td>ScopedBy</td>
<td>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.</td>
</tr>
<tr>
<td>RsConnector</td>
<td>A relation that associates a configuration item to an MConn.</td>
</tr>
<tr>
<td>DevCtx</td>
<td>Allows a configuration item to be associated with a specific physical device (CDev) in a device (LDev).</td>
</tr>
<tr>
<td>Locked</td>
<td>Allows a configuration item value to be locked. Once locked, the value cannot be changed.</td>
</tr>
</tbody>
</table>

MPARAM

MPARAM is the basic unit of configuration parameters that declares a single configuration parameter. MPARAM has following attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>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.</td>
</tr>
<tr>
<td>Description</td>
<td>Describes the configuration item.</td>
</tr>
</tbody>
</table>
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:

```xml
<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 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>
```

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.
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:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>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.</td>
</tr>
<tr>
<td>Description</td>
<td>Describes the configuration item.</td>
</tr>
<tr>
<td>Cardinality</td>
<td>Specifies the cardinality of the configuration item. The default value is 1.</td>
</tr>
<tr>
<td>ScopedBy</td>
<td>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.</td>
</tr>
<tr>
<td>DevCtx</td>
<td>Allows a configuration item to be associated with a specific physical device (CDev) in a device cluster (LDev).</td>
</tr>
<tr>
<td>Locked</td>
<td>Allows a configuration item value to be locked. Once locked, the value cannot be changed.</td>
</tr>
</tbody>
</table>

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:
### Configuration Parameters

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>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.</td>
</tr>
<tr>
<td>Value</td>
<td>Holds the value for a given configuration item. Values are not supported for MPParam.</td>
</tr>
<tr>
<td>Description</td>
<td>Describes the configuration item.</td>
</tr>
<tr>
<td>Cardinality</td>
<td>Specifies the cardinality of the configuration item. The default value is 1.</td>
</tr>
<tr>
<td>Mandatory</td>
<td>Allows a configuration item to be marked as mandatory.</td>
</tr>
<tr>
<td>Locked</td>
<td>Allows a configuration item value to be locked. Once locked, the value cannot be changed.</td>
</tr>
<tr>
<td>Validation</td>
<td>Specifies the validation mechanism to be used to validate the configuration parameter.</td>
</tr>
</tbody>
</table>

**AbsRel**

AbsRel allows one AbsFolder to refer to another AbsFolder. An AbsRel has following attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>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.</td>
</tr>
<tr>
<td>Value</td>
<td>Holds the value for a given configuration item. Values are not supported for MPParam.</td>
</tr>
<tr>
<td>Description</td>
<td>Describes the configuration item.</td>
</tr>
<tr>
<td>Cardinality</td>
<td>Specifies the cardinality of the configuration item. The default value is 1.</td>
</tr>
<tr>
<td>Mandatory</td>
<td>Allows a configuration item to be marked as mandatory.</td>
</tr>
</tbody>
</table>

**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:

```xml
<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 AbsParams 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:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>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.</td>
</tr>
<tr>
<td>Description</td>
<td>Describes the configuration item.</td>
</tr>
<tr>
<td>Cardinality</td>
<td>Specifies the cardinality of the configuration item. The default value is $1$.</td>
</tr>
<tr>
<td>ScopedBy</td>
<td>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 $Epg$.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>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.</td>
</tr>
<tr>
<td>Value</td>
<td>Holds the value for a given configuration item. Values are not supported for MParam.</td>
</tr>
<tr>
<td>Description</td>
<td>Describes the configuration item.</td>
</tr>
<tr>
<td>Cardinality</td>
<td>Specifies the cardinality of the configuration item. The default value is $1$.</td>
</tr>
</tbody>
</table>

RsCfgToConn   | A relation that associates a configuration item to an MConn.             |
### AbsRel

AbsRel allows one AbsFolder to refer to another AbsFolder. AbsRel has the following attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>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.</td>
</tr>
<tr>
<td>Value</td>
<td>Holds the value for a given configuration item. Values are not supported for MParam.</td>
</tr>
<tr>
<td>Description</td>
<td>Describes the configuration item.</td>
</tr>
<tr>
<td>Cardinality</td>
<td>Specifies the cardinality of the configuration item. The default value is 1.</td>
</tr>
<tr>
<td>RsCfgToConn</td>
<td>A relation that associates a configuration item to an MConn.</td>
</tr>
<tr>
<td>Mandatory</td>
<td>Allows a configuration item to be marked as mandatory.</td>
</tr>
<tr>
<td>Locked</td>
<td>Allows a configuration item value to be locked. Once locked, the value cannot be changed.</td>
</tr>
</tbody>
</table>

### Example XML POST for an Abstract Function Node With Configuration Parameters

The following XML POST example shows an abstract function node with configuration parameters:

```xml
<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>
  </vnsAbsDevCfg>
</vnsAbsNode>
```
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.
FolderInst

FolderInst is a group of configuration items that can contain ParamInsts and other nested FolderInsts. A FolderInst has the following attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>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.</td>
</tr>
<tr>
<td>ctrctNameOrLbl</td>
<td>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 &quot;any&quot; to allow this FolderInst to be used for all contracts.</td>
</tr>
<tr>
<td>graphNameOrLbl</td>
<td>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.</td>
</tr>
<tr>
<td>nodeNameOrLbl</td>
<td>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.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>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.</td>
</tr>
<tr>
<td>Value</td>
<td>Holds the value for a given configuration item. Values are not supported for MParam.</td>
</tr>
</tbody>
</table>

CfgRelInst

CfgRelInst has following attributes:

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Key</td>
<td>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.</td>
</tr>
<tr>
<td>Value</td>
<td>Holds the path for the target FolderInst.</td>
</tr>
</tbody>
</table>

Example XML POST for an Application EPG With Configuration Parameters

The following XML example shows configuration parameters inside of the device package:

```xml
<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="G2" 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"..."
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.
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**

```
+----------------------------------+
| Tenant                           |
| +----------------+---------------+|
| | App Profile    | L3Ctx         |
| | +-------+-----+ | +-------+-----+|
| | EPg    | BD   |               |
| +----------------+---------------+|
```

### 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:

```java
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

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>aaa:Boolean</code></td>
<td>scalar:Enum8</td>
<td><code>allowWrites</code></td>
<td>Read-write or read rule.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>(aaa:RbacRule:allowWrites)</td>
<td></td>
</tr>
</tbody>
</table>
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:

```plaintext
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

<table>
<thead>
<tr>
<th>Property</th>
<th>Type</th>
<th>Class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>aaa:Boolean</code></td>
<td>scalar:Enum8</td>
<td>allowWrites (aaa:PartialRbacRule:allowWrites)</td>
<td>Read-write or read rule.</td>
</tr>
<tr>
<td><code>naming:Name</code></td>
<td>string:Basic</td>
<td>domain (aaa:PartialRbacRule:domain)</td>
<td>The domain of the counts object.</td>
</tr>
<tr>
<td><code>reference:BinRef</code></td>
<td></td>
<td>monPolDn (aaa:PartialRbacRule:monPolDn)</td>
<td>The monitoring policy attached to this observable object.</td>
</tr>
<tr>
<td><code>reference:BinRef</code></td>
<td></td>
<td>partialObjectDn (aaa:PartialRbacRule:partialObjectDn)</td>
<td></td>
</tr>
</tbody>
</table>

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 `RbacRules` that point to any distinguished name in the system. The tenant administrator can only create `PartialRbacRules` 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 `PartialRbacRules` that grant access to specific firewall and load balancer devices to their own tenant users. The access is implemented by creating `aaaDomains` that represent each resource group that must be individually assigned. The following information provides an example setup:

<table>
<thead>
<tr>
<th>Tenant</th>
<th>Acme</th>
</tr>
</thead>
</table>
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 PartialRbacRules 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 15

Using a Service Graph Template

- Associating Service Graph Templates with Contracts and EPGs Using the GUI, on page 165
- Creating a Service Graph Template Using the NX-OS-Style CLI, on page 165
- Configuring a Service Graph Template Using the REST APIs, on page 168

**Associating Service Graph Templates with Contracts and EPGs Using the GUI**

See Using the GUI, on page 193 for the procedure for associating service graph templates with contracts and EPGs.

**Creating a Service Graph Template Using the NX-OS-Style CLI**

The following procedure creates a service graph template.

**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**
Associate a service graph to the template.

```
l4l7_graph graph_name contract contract_name
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>graph</td>
<td>The name of the service graph template.</td>
</tr>
<tr>
<td>contract</td>
<td>The name of the contract to use with the service graph template.</td>
</tr>
</tbody>
</table>
### Example:
```
apic1(config-tenant)## 1417 graph GraphL3asa contract ContractL3ASA
```

#### Step 4
Add a function node.
```
service node_name [device-cluster-tenant tenant_name] [device-cluster device_name] [mode deployment_mode]
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>service</td>
<td>The name of the service node to add.</td>
</tr>
<tr>
<td>device-cluster-tenant</td>
<td>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.</td>
</tr>
<tr>
<td>device-cluster</td>
<td>Name of the device cluster to use for this service node.</td>
</tr>
<tr>
<td>mode</td>
<td>The deployment mode. Possible values are:</td>
</tr>
<tr>
<td></td>
<td>• ADC_ONE_ARM—Specifies one-arm mode.</td>
</tr>
<tr>
<td></td>
<td>• ADC_TWO_ARM—Specifies two-arm mode.</td>
</tr>
<tr>
<td></td>
<td>• FW_ROUTED—Specifies routed (GoTo) mode.</td>
</tr>
<tr>
<td></td>
<td>• FW_TRANS—Specifies transparent (GoThrough) mode.</td>
</tr>
<tr>
<td></td>
<td>• OTHERS</td>
</tr>
<tr>
<td></td>
<td>If the mode is not specified, then a deployment mode is not used.</td>
</tr>
</tbody>
</table>

#### Example:
```
apic1(config-graph)## service Node1 device-cluster-tenant common device-cluster ifav108-asa-2 mode FW_ROUTED
```

#### Step 5
Add the consumer connector.
```
connector connector_type [cluster-interface interface_type]
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>connector</td>
<td>The type of the connector in the service graph. Possible values are:</td>
</tr>
<tr>
<td></td>
<td>• provider</td>
</tr>
<tr>
<td></td>
<td>• consumer</td>
</tr>
<tr>
<td>cluster-interface</td>
<td>The type of the device cluster interface. Possible values are:</td>
</tr>
<tr>
<td></td>
<td>• provider</td>
</tr>
<tr>
<td></td>
<td>• consumer</td>
</tr>
<tr>
<td></td>
<td>Do not specify this parameter if you are a service graph template in tenant Common.</td>
</tr>
</tbody>
</table>

#### Example:
```
apic1(config-service)## connector consumer cluster-interface consumer
```
Step 6

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
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tenant</td>
<td>The name of the tenant to associate with the connector.</td>
</tr>
<tr>
<td>out</td>
<td>The name of the Layer 3 outside.</td>
</tr>
<tr>
<td>epg</td>
<td>The name of the endpoint group.</td>
</tr>
<tr>
<td>redistribute</td>
<td>The properties of the redistribute protocol.</td>
</tr>
</tbody>
</table>

Example:

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

Step 7

Repeat steps 5 and 6 for the provider.

Example:

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

Step 8

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

```
rtr-cfg router_ID
  exit
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>rtr-cfg</td>
<td>The ID of the router.</td>
</tr>
</tbody>
</table>

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

Example:

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

Step 9

Associate a connection with a consumer connector and another with a provider connector, and then exit the service graph configuration mode.

```
  connection connection_name terminal terminal_type service node_name
      connector connector_type
  exit
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>connection</td>
<td>The name of the connection to associate with the connector.</td>
</tr>
<tr>
<td>terminal</td>
<td>The type of the terminal. Possible values are:</td>
</tr>
<tr>
<td></td>
<td>• provider</td>
</tr>
<tr>
<td></td>
<td>• consumer</td>
</tr>
</tbody>
</table>

Creating a Service Graph Template Using the NX-OS-Style CLI
### Configuring a Service Graph Template Using the REST APIs

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

```xml
<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"/>
    </fvBD>
    <!-- Bridge Domain for MyClnt EPG -->
    <fvBD name="MyClntBD">
      <fvRsCtx tnFvCtxName="MyNetwork"/>
      <fvSubnet ip="20.20.20.20/24"/>
    </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"/>
        <fvRsPathAtt tDn="topology/pod-1/paths-17/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"/>
    </fvAEPg>
  </fvTenant>
</polUni>
```

### Configuring a Service Graph Template Using the REST APIs

#### Example:

```bash
apic1(config-graph)# connection C1 terminal consumer service Node1 connector consumer
apic1(config-graph)# connection C2 terminal provider service Node1 connector provider
apic1(config-graph)# exit
```

#### Step 10

Exit the configuration mode.

#### Example:

```bash
apic1(config-tenant)# exit
apic1(config)# exit
```
Creating a Security Policy Using the REST APIs

You can create a security policy using the following REST API:

```xml
<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>
```
Monitoring a Service Graph

- Monitoring a Service Graph Instance Using the GUI, on page 171
- Monitoring Service Graph Faults Using the GUI, on page 172
- Resolving Service Graph Faults, on page 172
- Monitoring a Virtual Device Using the GUI, on page 177
- Monitoring Device Cluster and Service Graph Status Using the NX-OS-Style CLI, on page 178

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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Service Graph column</td>
<td>The name of the service graph template.</td>
</tr>
<tr>
<td>Contract column</td>
<td>The name of the contract that is shown in the service graph template.</td>
</tr>
<tr>
<td>Contained By column</td>
<td>The name of the network that contains the service graph template.</td>
</tr>
<tr>
<td>State column</td>
<td>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.</td>
</tr>
<tr>
<td>Description column</td>
<td>The description of the service graph.</td>
</tr>
</tbody>
</table>

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:

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>POLICY tab</td>
<td>The function's properties, resources allocated to the function, and the parameters of the function.</td>
</tr>
<tr>
<td>FAULTS tab</td>
<td>The issues that are happening on the function node.</td>
</tr>
<tr>
<td>HISTORY tab</td>
<td>The history of events that occurred on the function node.</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Fault</th>
<th>CLI Label</th>
<th>Description and Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>missing-connection</td>
<td>connection associated with a connector not found</td>
<td>The configuration for a graph connector is invalid. The associated connection for the connector could not be found.</td>
</tr>
</tbody>
</table>
### Table 5: AbsGraph and GraphInst Faults

<table>
<thead>
<tr>
<th>Fault</th>
<th>CLI Label</th>
<th>Description and Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>missing-nodeinst</td>
<td>NodeInst associated with a</td>
<td>The configuration for a graph connector is invalid. The associated NodeInst for the connector could not be found.</td>
</tr>
<tr>
<td></td>
<td>connector not found</td>
<td></td>
</tr>
<tr>
<td>conn-nonrenderable</td>
<td>Graph connector could not be</td>
<td>The configuration for a graph connector is invalid. The graph could not be rendered.</td>
</tr>
<tr>
<td></td>
<td>rendered.</td>
<td></td>
</tr>
<tr>
<td>invalid-bd</td>
<td>BD associated with a connector is not valid</td>
<td>The configuration for a graph connector is invalid. The associated bridge domain for the connector is not valid.</td>
</tr>
<tr>
<td>invalid-ctx</td>
<td>Ctx associated with a connector is not valid.</td>
<td>The configuration for a graph connector is invalid. The associated Ctx for the connector is not valid.</td>
</tr>
<tr>
<td>missing-peer-conn</td>
<td>Peer connector associated with a connector not found.</td>
<td>Configuration for a graph connector is invalid. The peer connector for the connection could not be found.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Fault</th>
<th>CLI Label</th>
<th>Description and Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>invalid-abstract-graph-config</td>
<td>invalid abstract graph config</td>
<td>The abstract graph configuration is invalid.</td>
</tr>
<tr>
<td>missing-mandatory-param</td>
<td>mandatory param not found</td>
<td>A mandatory configuration parameter could not be resolved. Check the package for the mandatory parameter and make sure that AbsGraph has the parameter.</td>
</tr>
<tr>
<td>param-cardinality-error</td>
<td>invalid param cardinality</td>
<td>A configuration parameter does not meet cardinality requirements. Check if you specified multiple instances of a parameter without specifying cardinality=n.</td>
</tr>
<tr>
<td>epp-download-failure</td>
<td>epp download failure</td>
<td>Graph policies failed to download to the switch.</td>
</tr>
<tr>
<td>param-duplicate-name-failure</td>
<td>duplicate param name</td>
<td>Multiple identical copies of a parameter were found with the same name.</td>
</tr>
<tr>
<td>id-allocation-failure</td>
<td>id allocation failure</td>
<td>A unique network resource (either VLAN or VXLAN) could not be allocated.</td>
</tr>
<tr>
<td>missing-ldev</td>
<td>No cluster found</td>
<td>A cluster could not be found.</td>
</tr>
<tr>
<td>Fault</td>
<td>CLI Label</td>
<td>Description and Resolution</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>-----------------------------------------------</td>
<td>-------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>context-cardinality-violation-failure</td>
<td>invalid cluster context cardinality</td>
<td>The cluster does not support the required tenancy (multi-tenant or single tenant).</td>
</tr>
<tr>
<td>function-type-mismatch-failure</td>
<td>invalid function type</td>
<td>The function type is not supported for the selected device. Check if the AbsNode functype and resolved LDevVip function type match.</td>
</tr>
<tr>
<td>invalid-abstract-graph-config-param</td>
<td>invalid abstract graph config param</td>
<td>The abstract graph configuration parameter is invalid.</td>
</tr>
<tr>
<td>missing-mparam</td>
<td>No parameter definition found</td>
<td>A required parameter definition could not be found.</td>
</tr>
<tr>
<td>missing-abs-graph</td>
<td>no abs graph found</td>
<td>The abstract graph configuration is missing for the graph instance.</td>
</tr>
<tr>
<td>invalid-param-config</td>
<td>invalid param config</td>
<td>The parameter configuration is invalid.</td>
</tr>
<tr>
<td>invalid-param-scope</td>
<td>invalid parameter scope</td>
<td>The parameter scope is invalid. Check the vnsRsScopeToTerm parameter in the AbsGraph to see if parameter is correct.</td>
</tr>
<tr>
<td>invalid-ldev</td>
<td>Invalid cluster</td>
<td>The cluster configuration is invalid. Check the status of the resolved LDevVip and correct the fault.</td>
</tr>
<tr>
<td>missing-tenant</td>
<td>no tenant found</td>
<td>The tenant could not be found for the graph.</td>
</tr>
<tr>
<td>internal-error</td>
<td>internal error</td>
<td>An internal error occurred during graph processing.</td>
</tr>
<tr>
<td>resource-allocation-failure</td>
<td>resource allocation failure</td>
<td>A required resource could not be allocated during graph processing.</td>
</tr>
<tr>
<td>missing-abs-function</td>
<td>no abstract function found</td>
<td>The abstract function definition is missing.</td>
</tr>
<tr>
<td>param-validation-failed</td>
<td>param validation failure</td>
<td>The configuration parameter value is invalid.</td>
</tr>
<tr>
<td>missing-mconn</td>
<td>No connector found</td>
<td>A required connector could not be found.</td>
</tr>
<tr>
<td>cdev-missing-mgmt-ip</td>
<td>no mgmt ip found for cdev</td>
<td>The management IP address could not be found for concrete device. Check if vnsCMgmt is present for the resolved vnsCDev.</td>
</tr>
<tr>
<td>invalid-graphinst</td>
<td>invalid graphinst config</td>
<td>The graph instance is invalid.</td>
</tr>
<tr>
<td>missing-interface</td>
<td>no interface found</td>
<td>An interface could not be found.</td>
</tr>
<tr>
<td>Fault</td>
<td>CLI Label</td>
<td>Description and Resolution</td>
</tr>
<tr>
<td>-----------------------</td>
<td>-------------------------</td>
<td>------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>missing-bd</td>
<td>no bd found</td>
<td>A bridge domain could not be found.</td>
</tr>
<tr>
<td>missing-terminal</td>
<td>Terminal node is missing a terminal</td>
<td>Terminal node is missing a terminal. Check the terminal node settings.</td>
</tr>
<tr>
<td>missing-namespace</td>
<td>no vlan/vxlan namespace found</td>
<td>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.</td>
</tr>
<tr>
<td>missing-mfunc</td>
<td>No function found in device package</td>
<td>A required function could not be found in the device package. Verify that all AbsNode function types are present in the package.</td>
</tr>
<tr>
<td>missing-lif</td>
<td>no cluster interface found</td>
<td>A required cluster interface could not be found. Verify that the vnsLIf parameter in vnsLDevVip is configured correctly.</td>
</tr>
<tr>
<td>invalid-absfunc-profile</td>
<td>Abstract Function Profile config is invalid</td>
<td>The abstract function profile configuration is invalid. This fault may be due to an invalid configuration parameter that is specified in the profile.</td>
</tr>
<tr>
<td>missing-cdev</td>
<td>No device found</td>
<td>The concrete device could not be found in the cluster. Verify that a valid vnsCDev is present under the resolved vnsLDevVip.</td>
</tr>
<tr>
<td>inappropriate-devfolder</td>
<td>Illegal folder in configuration</td>
<td>No corresponding folder was found in the device package.</td>
</tr>
<tr>
<td>invalid-devctx</td>
<td>Device context is not legal for this folder</td>
<td>The device package does not allow specifying a device context for this folder.</td>
</tr>
<tr>
<td>insufficient-devctx</td>
<td>Folder must have one value for each associated CDev</td>
<td>The folder is concrete device specific. The folder must have at least one value for each concrete device.</td>
</tr>
<tr>
<td>cdev-missing-cif</td>
<td>No interface defined</td>
<td>A concrete device must have at least one interface defined.</td>
</tr>
<tr>
<td>Fault</td>
<td>CLI Label</td>
<td>Description and Resolution</td>
</tr>
<tr>
<td>--------------------------</td>
<td>---------------------------</td>
<td>---------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>cdev-missing-pathinfo</td>
<td>Missing path for interface</td>
<td>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.</td>
</tr>
<tr>
<td>missing-cif</td>
<td>Device interfaces does not match cluster</td>
<td>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.</td>
</tr>
<tr>
<td>ldevvip-missing-mgmt-ip</td>
<td>No Mgmt ip found for LDevVip</td>
<td>The management IP address could not be found for LDevVip.</td>
</tr>
<tr>
<td>lif-invalid-MIf</td>
<td>LIf has an invalid MIfLbl</td>
<td>The MIfLbl contained by LIf is not present in the device package.</td>
</tr>
<tr>
<td>lif-invalid-CIf</td>
<td>LIf has an invalid CIf</td>
<td>The CIf contained by LIf is not present. Check the concrete device and CIf settings.</td>
</tr>
<tr>
<td>missing-function-node</td>
<td>Abstract graph missing function node</td>
<td>An abstract graph must have at least one function node.</td>
</tr>
<tr>
<td>graph-loop-detected</td>
<td>Abstract graph config has a loop</td>
<td>The abstract graph configuration is invalid. The configuration has a loop.</td>
</tr>
<tr>
<td>gothrough-routing-enabled-both</td>
<td>Both the legs of go through node has routing enabled</td>
<td>Both the legs of the go through node have routing enabled.</td>
</tr>
<tr>
<td>invalid-terminal-nodes</td>
<td>Abstract graph has invalid number of terminal nodes</td>
<td>An abstract graph must have at least two terminal nodes.</td>
</tr>
<tr>
<td>missing-ldev-ctx</td>
<td>No device context found for LDev</td>
<td>The device context for the device could not be found. Verify that vnsLDevCtx has values that match the contract, graph and node.</td>
</tr>
<tr>
<td>arp-flood-enabled</td>
<td>ARP flood is enabled on the management endpoint group</td>
<td>ARP flood must be disabled for the management endpoint group.</td>
</tr>
<tr>
<td>folderinst-validation-failed</td>
<td>FolderInst has key, that is not found in MFolder</td>
<td>The FolderInst’s key and value should honor MFolder specifications.</td>
</tr>
<tr>
<td>paraminst-validation-failed</td>
<td>ParamInst has key and/or value, that are not found in MParam</td>
<td>ParamInst’s key and value should honor MParam specifications.</td>
</tr>
<tr>
<td>invalid-mfolder</td>
<td>FolderInst points to an invalid MFolder</td>
<td>FolderInst must point to a valid MFolder.</td>
</tr>
</tbody>
</table>
### 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:

<table>
<thead>
<tr>
<th>Tab</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>POLICY tab</td>
<td>The device that is in use, the VLANs that are configured within the device, and the parameters that have been passed to the devices.</td>
</tr>
<tr>
<td>OPERATIONAL tab</td>
<td>The statistics that are being received from the various devices.</td>
</tr>
<tr>
<td>HEALTH tab</td>
<td>The health of the devices.</td>
</tr>
</tbody>
</table>

---

<table>
<thead>
<tr>
<th>Fault</th>
<th>CLI Label</th>
<th>Description and Resolution</th>
</tr>
</thead>
<tbody>
<tr>
<td>invalid-mparam</td>
<td>ParamInst points to an invalid MParam.</td>
<td>ParamInst must point to a valid MParam.</td>
</tr>
<tr>
<td>devfolder-validation-failed</td>
<td>DevFolder has key, that is not found in MFolder</td>
<td>DevFolders key and value should honor MFolder specifications.</td>
</tr>
<tr>
<td>devparam-validation-failed</td>
<td>DevParam has key and/or value, that are not found in MParam</td>
<td>DevParam's key and value should honor MParam specifications.</td>
</tr>
<tr>
<td>cdev-missing-virtual-info</td>
<td>Virtual Object Info is missing in CDev</td>
<td>Virtual object information must be provided if LDevVip is of type Virtual.</td>
</tr>
<tr>
<td>invalid-rsmconnatt</td>
<td>Relationship to metaconnector is invalid</td>
<td>Correct the metaconnector DN and ensure it binds to the correct MDev hierarchy.</td>
</tr>
</tbody>
</table>
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:
show 1417-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 1417-cluster

Example:

apic1# show faults 1417-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 1417-graph

Example:

apic1# show faults 1417-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 1417 cluster
```

Example:
```
apic1# show running-config tenant common 1417 cluster
# Command: show running-config tenant common 1417 cluster
# Time: Thu Nov 26 00:35:59 2015
tenant common
  1417 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
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 1417 graph
```

Example:
```
apic1# show running-config tenant common 1417 graph
# Command: show running-config tenant common 1417 graph
# Time: Thu Nov 26 00:35:59 2015
tenant T1
  1417 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
```
Configuring Administrator Roles for Managing a Service Configuration

• About Privileges, on page 181
• Configuring a Role for Device Management, on page 182
• Configuring a Role for Service Graph Template Management, on page 182
• Configuring a Role for Uploading Device Package, on page 182
• Configuring a Role for Exporting Devices, on page 182

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:

<table>
<thead>
<tr>
<th>Privilege</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>nw-svc-policy</td>
<td>The network service policy privilege enables you to do the following:</td>
</tr>
<tr>
<td></td>
<td>• Create a service graph template</td>
</tr>
<tr>
<td></td>
<td>• Attach a service graph template to an application endpoint group (EPG) and a contract</td>
</tr>
<tr>
<td></td>
<td>• Monitor a service graph</td>
</tr>
<tr>
<td>nw-svc-device</td>
<td>The network service device privilege enables you to do the following:</td>
</tr>
<tr>
<td></td>
<td>• Create a device</td>
</tr>
<tr>
<td></td>
<td>• Create a concrete device</td>
</tr>
<tr>
<td></td>
<td>• Create a device context</td>
</tr>
</tbody>
</table>

Note

Only the infrastructure administrator can upload a device package to the 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 templates, 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.
Developing Automation

- About the REST APIs, on page 183
- Examples of Automating Using the REST APIs, on page 183

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 APIC UI 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 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:
The following REST request creates a VLAN namespace:

```
<infraInfra>
  <fvnsVlanInstP name="MyNS" allocMode="dynamic">
    <fvnsEncapBlk name="encap" from="vlan-201" to="vlan-300"/>
  </fvnsVlanInstP>
</infraInfra>
```

The following REST request creates a VMM domain:

```
<vmmProvP vendor="Vendor1">
  <vmmDomP name="MyVMs">
    <infraRsVlanNs tDn="uni/infra/vlanns-MyNS-dynamic"/>
  </vmmDomP>
</vmmProvP>
```
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:

```
<fvTenant dn="uni/tn-acme" name="acme">
  <vnsLDevVip name="ADCCluster1" contextAware=1>
    <vnsRsMDevAtt tDn="uni/infra/mDev-Acme-ADC-1.0"/>
    <vnsRsDevEp 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>
```

The following REST request creates an unmanaged device cluster:

```
<fvTenant name="HA_Tenant1">
  <vnsLDevVip name="ADCCluster1" devtype="VIRTUAL" managed="no">
    <vnsRsALDevToDomP tDn="uni/vmmp-VMware/dom-mininet"/>
  </vnsLDevVip>
</fvTenant>
```

The following REST request creates a device cluster context:

```
<fvTenant dn="uni/tn-acme" name="acme">
  <vnsLDevCtx ctcrtNameOrLbl="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>
```

The following REST request creates a device cluster context used in route peering:

```
<fvTenant dn="uni/tn-coke{{tenantId}}" name="coke{{tenantId}}">
  <vnsRtrCfg name="Dev1Ctx1" rtrId="180.0.0.12">
    <vnsLDevCtx ctcrtNameOrLbl="webCtxt" graphNameOrLbl="WebGraph" nodeNameOrLbl="FW">
      <vnsRsLDevCtxToLDev tDn="uni/tn-tenant1/lDevVip-Firewall"/>
      <vnsRsLDevCtxToRtrCfg tnVnsRtrCfgName="FwRtrCfg"/>
    </vnsLDevCtx>
  </vnsRtrCfg>
</fvTenant>
```
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:

```xml
<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"/>
        <vnsLIf/>
      </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-int"/>
      </vnsLIf>
    </vnsLDevVip>
  </fvTenant>
</polUni>
```

The following REST request adds a concrete device in a physical device cluster:

```xml
<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"/>
        <vnsCCred 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>
```

---

### 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:

```xml
<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"/>
        <vnsLIf/>
      </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-int"/>
      </vnsLIf>
    </vnsLDevVip>
  </fvTenant>
</polUni>
```

The following REST request adds a concrete device in a physical device cluster:

```xml
<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"/>
        <vnsCCred 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>
```
The following REST request adds a concrete device in a virtual device cluster:

```xml
<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"/>
<vnsCCred name="password" value="admin"/>
</vnsCDev>
</fvTenant>
</polUni>
```

The following REST request creates a service graph in managed mode:

```xml
<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>
</fvTenant>
</polUni>

The following REST request creates a service graph in managed mode:

```xml
<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/outtmnl1"/>
    <vnsAbsFuncConn name = "C4" direction = "input">
      <vnsRsMConnAtt tDn="uni/infra/mDev-Acme-ADC-1.0/mFunc-SLB/mConn-external"/>
    </vnsAbsFuncConn>
    <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>
</vnsAbsGraph>
</fvTenant>
</polUni>
```
The following REST request creates a service graph in unmanaged mode:

```xml
<polUni>
  <fvTenant name="HA_Tenant1">
    <vnsAbsGraph name="g1">
      <vnsAbsTermNodeProv name="Input1">
        <vnsAbsTermConn name="C1">
          ...<vnsAbsTermConn>
        </vnsAbsTermConn>
        <!-- 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>
          <vnsAbsFuncConn name="inside" attNotify="true">
            ...<vnsAbsFuncConn>
          </vnsAbsFuncConn>
        </vnsAbsNode>
      </vnsAbsTermNodeProv>
      <vnsAbsTermNodeCon name="Output1">
        <vnsAbsTermConn name="C6">
          ...<vnsAbsTermConn>
        </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 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"/>

        <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"/>

        <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="192.168.100.100"/>
          <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="false"/>
          <vnsParamInst name="cip" key="cip" value="ENABLED"/>
          <vnsParamInst name="cxa" key="cxa" value="NO"/>
          <vnsParamInst name="sp" key="sp" value="OFF"/>
        </vnsFolderInst>
  </fvAp>

</fvTenant>
```
The following REST request attaches a service graph to a contract:

```xml
<poUnl>
  <fvTenant name="acme">
    <vzBrCP name="webCtrct">
      <vzSubj name="http">
        <vzRsSubjGraphAtt graphName="G1" termNodeName="Input1"/>
      </vzSubj>
    </vzBrCP>
  </fvTenant>
</poUnl>
```
Developing Automation

Examples of Automating Using the REST APIs

</v2BrCP>
</fvTenant>
</polUni>
Deploying the Layer 4 to Layer 7 Services Using the GUI

You can deploy the Layer 4 to Layer 7 services using GUI. Perform the procedures in the following order:

1. Import a device package.
   
   See Importing a Device Package Using the GUI, on page 194.

2. Create a function profile.
   
   See Creating a Function Profile Using the GUI, on page 194.

3. Create a service graph template.
   
   See Creating a Layer 4 to Layer 7 Service Graph Template Using the GUI, on page 196.

4. Create a device.
   
   See Creating a Layer 4 to Layer 7 Device Using the GUI, on page 15.
   
   (Optional) Modify a device.
   
   See Modifying a Device, on page 197.

5. Apply a service graph template to endpoint groups (EPGs).
   
   See Applying a Service Graph Template to Endpoint Groups Using the GUI, on page 198.
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 APIC what devices you have and what the devices can do.

Step 1
Download an appropriate device package. You can find the list of partners at the following URL:
This URL is the Partner Ecosystem page, where you can download the appropriate device package.

Step 2
Log in to the 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.

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.
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.

Creating a Layer 4 to Layer 7 Service Graph Template Using the GUI

A service graph template is a sequence of Layer 4 to Layer 7 functions, Layer 4 to Layer 7 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 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 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 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 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.

---

## Modifying a Device

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.
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.