Cisco APIC Layer 4 to Layer 7 Services Deployment Guide

First Published: October 31, 2013

Last Modified: September 29, 2015

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

CONTENTS

Preface

This preface includes the following sections:

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
- Server administration
- Switch and network administration

Document Conventions

Command descriptions use the following conventions:

Convention	Description
bold	Bold text indicates the commands and keywords that you enter literally as shown.
Italic	Italic text indicates arguments for which the user supplies the values.
[x]	Square brackets enclose an optional element (keyword or argument).

Convention	Description
[x | y]	Square brackets enclosing keywords or arguments separated by a vertical bar indicate an optional choice.
{x | y}	Braces enclosing keywords or arguments separated by a vertical bar indicate a required choice.
[x {y | z}]	Nested set of square brackets or braces indicate optional or required choices within optional or required elements. Braces and a vertical bar within square brackets indicate a required choice within an optional element.
`variable`	Indicates a variable for which you supply values, in context where italics cannot be used.
string	A nonquoted set of characters. Do not use quotation marks around the string or the string will include the quotation marks.

Examples use the following conventions:

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

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

The Application Centric Infrastructure documentation set includes the following documents:

Web-Based Documentation

- *Cisco APIC Management Information Model Reference*
- *Cisco APIC Online Help Reference*
- *Cisco ACI MIB Support List*

Downloadable Documentation

- *Cisco Application Centric Infrastructure Release Notes*
- *Cisco Application Centric Infrastructure Fundamentals Guide*
- *Cisco APIC Getting Started Guide*
- *Cisco APIC REST API User Guide*
- *Cisco APIC Command Line Interface User Guide*
- *Cisco APIC Faults, Events, and System Message Guide*
- *Cisco APIC Layer 4 to Layer 7 Device Package Development Guide*
- *Cisco APIC Layer 4 to Layer 7 Services Deployment Guide*
- *Cisco ACI Firmware Management Guide*
- *Cisco ACI Troubleshooting Guide*
- *Cisco ACI NX-OS Syslog Reference Guide*
- *Cisco ACI Switch Command Reference, NX-OS Release 11.0*
- *Cisco ACI MIB Quick Reference*
- *Cisco Nexus CLI to Cisco APIC Mapping Guide*
- *Installing the Cisco Application Virtual Switch with the Cisco APIC*
- *Configuring the Cisco Application Virtual Switch using the Cisco APIC*
- *Application Centric Infrastructure Fabric Hardware Installation Guide*

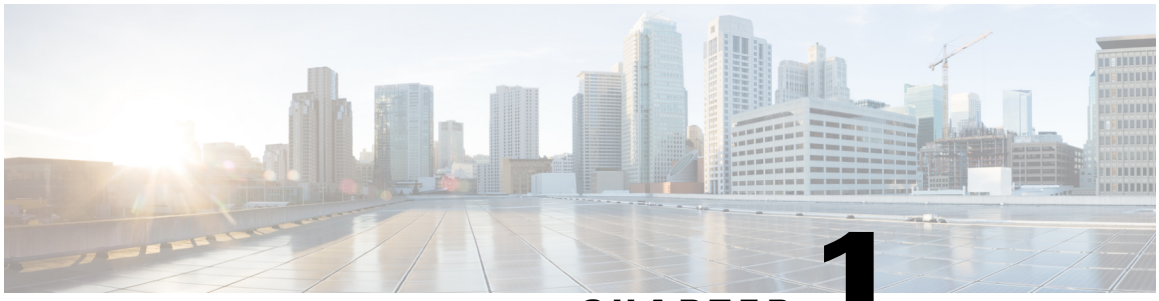
Documentation Feedback

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.



CHAPTER 1

Overview

Overview

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.

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

1. Import the device package

   Only the provider administrator can import the device package.

2. Configure a tenant
3. Register the device and the logical interfaces

   This task also registers concrete devices and concrete interfaces, and configures concrete device parameters.

4. Configure device (logical device) parameters
5. Configure a Layer 3 network
6. Configure a bridge domain
7. Configure an application profile

8. Configure a physical domain or a VMM domain

   For a VMM domain:

   a. Configure VMM domain credentials
   b. Configure a vCenter/vShield controller profile

9. Configure a VLAN pool

   a. Configure an encapsulation block range

10. Configure a contract
11. Configure a management endpoint group (EPG)
12. Configure a service graph template
13. Select the default service graph template parameters from an application profile
14. Configure the service graph template parameters, if needed
15. Attach the service graph template to a contract
16. Configure additional configuration parameters

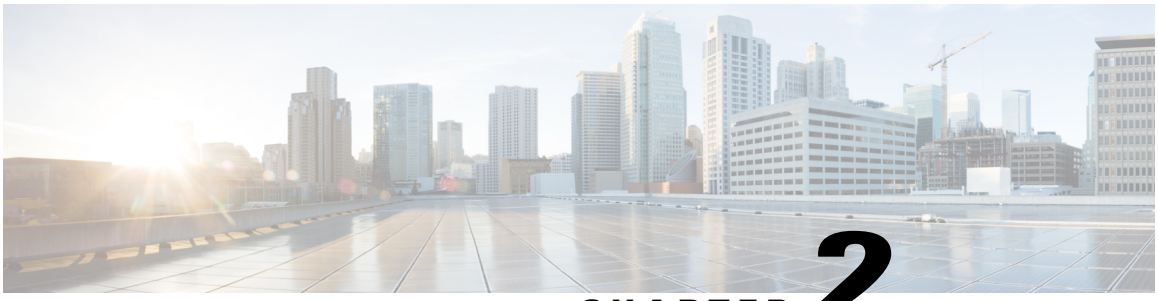
Note Virtualized appliances can be deployed with VLANs as the transport between VMware ESX servers and leaf nodes, and can be deployed only with VMware ESX as the hypervisor.

About 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. 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. 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.

ACI

The 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 abstract graph. When an abstract graph is referenced by a contract, the abstract graph 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 abstract graph. Another key concept is the "logical device", or device cluster, which represents the cluster of concrete devices. The rendering of the abstract graph is based on identifying the suitable logical devices that can be inserted in the path that is defined by a contract.



CHAPTER 2

Prerequisites

Prerequisites

About Tenants

A tenant is a container for policies that enable an administrator to exercise domain-based access control so that qualified users can access privileges, such as tenant administration and networking administration. You must configure a tenant before you can deploy any Layer 4 to Layer 7 services.

About Security Domains

A security domain is a concept that allows you to scope which tenant is accessible by which user. For example, if you create `Tenant1`, `Tenant2`, and `Tenant3`, you can create three security domains—`securitydomain1`, `securitydomain2`, and `securitydomain3`—and the administrators of each tenant would be associated with the respective security domain.

About Layer 3 Networks

Layer 3 is the network layer of the Open Systems Interconnection (OSI) communication model. An Layer 3 network configuration refers to the configuration of how traffic forwarding works to the outside of the fabric. Layer 3 is used to discover the address of other nodes, select routes, select quality of service, and forward incoming messages for local host domains to the transport layer. The Layer 3 network is used by all of the application endpoint groups (EPGs) that are used by the tenant.

About Bridge Domains

A bridge domain represents a Layer 2 forwarding construct within the fabric. One or more endpoint groups (EPGs) can be associated with one bridge domain or subnet. A bridge domain can have one or more subnets that are associated with it. One or more bridge domains together form a tenant network.

About Application Profiles

An application profile defines the policies, services and relationships between endpoint groups (EPGs). Each application profile contains one or more EPGs that can communicate with the other EPGs in the same application profile and with EPGs in other application profiles according to the contract rules.

About Contracts

A contract contains all of the filters that will be applied between endpoint groups (EPGs) that produce and consume the contract. A contract involves EPGs that are called providers and consumers. A contract defines the protocols and ports on which a provider and consumer are allowed to communicate.

Configuring a VLAN Pool

A VLAN pool is also known as a VLAN namespace. You can configure a VLAN pool.

Step 1 In the **CREATE VCENTER DOMAIN** dialog box, choose **Create VLAN Pool** from the **VLAN Pool** drop-down list. The **CREATE VLAN POOL** dialog box appears.

Step 2 Complete the following fields:

Name	Description
Name field	The name of the VLAN pool.
Description field	The description of the VLAN pool.
Allocation Mode radio buttons	The allocation mode of the VLAN pool. You can choose **Dynamic Allocation** or **Static Allocation**. **Note** Choose**Dynamic Allocation** if VMM needs to be integrated when devices are virtual.
Encap Blocks section	The encapsulation block ranges, which specify which VLANs to use while using a virtual appliance for performance graphs. To create an encapsulation block range, see #unique_27.

Step 3 Click **SUBMIT**. The **CREATE VLAN POOL** dialog box closes and the VLAN pool is created.

Configuring an Encapsulation Block Range

An encapsulation block range specifies which VLANs to use while using a virtual appliance for performance graphs. You can configure an encapsulation block range.

Step 1 In the **CREATE VLAN POOL** dialog box, click + in the **Encap Blocks** section. The **CREATE RANGES** dialog box appears.

Step 2 Complete the following fields:

Name	Description
From field	The minimum value for the encapsulation block range.
To field	The maximum value for the encapsulation block range.

Step 3 Click **OK**. The **CREATE RANGES** dialog box closes and the encapsulation block range is created.

Configuring a Physical Domain

Physical domains control the scope of where a given VLAN namespace is used. The VLAN namespace that is associated with the physical domain is for non-virtualized servers, although it can also be used for static mapping of port-groups from virtualized servers. You can configure a physical domain for physical device types.

Before You Begin

- Configure a tenant. See #unique_29.

Step 1 From the **Physical Domain** drop-down list, choose **Create Physical Domain**. The **CREATE PHYSICAL DOMAIN** dialog box appears.

Step 2 Complete the following fields:

Name	Description
Name field	The name of the physical domain profile.
VLAN Pool field	The VLAN pool of the physical domain. The VLAN pool specifies the range or pool for VLANs that is allocated by the APIC for the service graphs that are using this physical domain. To create a VLAN pool, see Configuring a VLAN Pool, on page 4.

Step 3 Click **Submit**. The **CREATE PHYSICAL DOMAIN** dialog box closes and the physical domain is created.

Configuring a VMM Domain

A Virtual Machine Manager (VMM) domain defines the scope of use of a given VLAN namespace for virtualized servers. A Virtual Machine Manager (VMM) domain is also called a vCenter domain. You can configure a VMM domain.

Before You Begin

- Configure a tenant. See #unique_29.
- Configure a device cluster on the tenant. See Configuring a Device Cluster.

Step 1 From the **VMM Domain** drop-down list, choose **Create vCenter Domain**. The **CREATE VCENTER DOMAIN** dialog box appears.

Step 2 Complete the following fields:

Name	Description
Name field	The name of the VMM domain profile.
Virtual Switch radio buttons	The mode of the virtual switch.
Associated Attachable Entity Profile field	The attachable entity profile that is to be associated with the VMM domain. The attachable entity profile is required to attach a VMM domain to the fabric.
VLAN Pool field	The VLAN pool of the VMM domain. The VLAN pool specifies the range or pool for VLANs that is allocated by the Application Policy Infrastructure Controller (APIC) for the service graphs that are using this VMM domain. To configure a VLAN pool, see Configuring a VLAN Pool, on page 4.
vCenter Credentials section	The credentials to use for connecting to the VMM domain. To configure vCenter credentials, which are also known as VMM credentials, see #unique_31.
vCenter/vShield section	The vCenter/vShield controller profile to use with the VMM domain. To configure a vCenter/vShield controller profile, see #unique_32.

Step 3 Click **OK**. The **CREATE VMM DOMAIN** dialog box closes and the VMM domain is created.

Configuring VMM Credentials

VMM credentials are required for connecting to the VMM domain. You can configure VMM credentials.

Step 1 In the **CREATE VCENTER DOMAIN** dialog box, click + in the **vCenter Credentials** section. The **CREATE VCENTER CREDENTIAL** dialog box appears.

Step 2 Complete the following fields:

Name	Description
Profile Name field	The name of the profile to use for logging into the VMM domain.
Description field	The description of the user account profile.
Username field	The name of the user to use for the credentials.
Password field	The password of the specified user.
Confirm Password field	The confirmation of the password of the specified user.

Step 3 Click **OK**. The **CREATE VCENTER CREDENTIAL** dialog box closes and the VMM credentials are created.

Configuring a vCenter/vShield Controller Profile

You can configure a vCenter/vShield controller profile.

Step 1 In the **CREATE VCENTER DOMAIN** dialog box, click + in the **vCenter/vShield** section. The **CREATE VCENTER/VSHIELD CONTROLLER** dialog box appears.

Step 2 Complete the following fields:

Name	Description
Type radio buttons	The profile type of the controller.
Name field	The name of the vCenter/vShield controller profile.
Address field	The hostname or IP address of the vCenter/vShield controller profile.
Stats Collection radio buttons	Enables or disables statistics collection.
Management EPG field and drop-down list	Choose the management endpoint group (EPG) in the Virtual Machine Manager (VMM) controller profile.
Associated Credential field and drop-down list	Choose the VMM credentials to use with the vCenter/vShield controller profile.

Step 3 Click **OK**. The **CREATE VCENTER/VSHIELD CONTROLLER** dialog box closes and the vCenter/vShield controller profile is created.

Configuring a Tenant

You can configure a tenant.

Step 1 On the menu bar, click the **TENANTS** tab. The **Tenant** window appears.

Step 2 On the submenu bar, click **ADD TENANT**. The **CREATE TENANT** dialog box appears, showing the **TENANT** page.

Step 3 Complete the following fields:

Name	Description
Name field	The name of the tenant.
Alias field	The alias for the tenant. The alias can be a simpler and more descriptive name than the tenant's name when referring to a single tenant. You can assign a particular alias name to only one tenant; the system will prevent you from assigning the same alias name to a second tenant.
Description field	The description of the tenant.
Tags field	A search keyword or term that is assigned to the tenant. A tag allows you to group multiple objects by a descriptive name. You can assign the same tag name to multiple objects and you can assign one or more tag names to an object.
Monitoring Policy field	The endpoint group (EPG) monitoring policy name.
Security Domains section	The security domains of the tenant. You do not need to choose a security domain to deploy Layer 4 to Layer 7 services. For information about creating a security domain, see the *Cisco APIC Getting Started Guide*.

Step 4 Click **Next**. The **NETWORK** page appears, and the tenant is created. To configure the Layer 3 network, see #unique_35.

Configuring a Bridge Domain

You can configure a bridge domain.

Before You Begin

- Configure a Layer 3 (L3) network. See #unique_35.

Step 1 On the **BRIDGE DOMAIN** page of the **CREATE TENANT** dialog box, complete the following fields:

Name	Description
Name field	The name of the bridge domain.
Description field	The description of the bridge domain.

Name	Description
Forwarding drop-down list	Choose the forwarding method of the bridge domain.
L2 Unknown Unicast radio buttons	The forwarding method for unknown Layer 2 destinations. These radio buttons appear only if you chose **Custom** in the **Forwarding** drop-down list.
Unknown Multicast Flooding radio buttons	Click **Flood** or **Optimized Flood**. These radio buttons appear only if you chose **Custom** in the **Forwarding** drop-down list.
Multi Destination Flooding radio buttons	Click **Flood in EPG**, **Drop**, or **Flood in BD**. These radio buttons appear only if you chose **Custom** in the **Forwarding** drop-down list.
ARP Flooding check box	Put a check in this box to enable ARP flooding. If flooding is disabled, unicast routing will be performed on the target IP address. This check box is unchecked by default. This check box appears only if you choose **Custom** in the **Forwarding** drop-down list.
GARP-Based Detection check box	Put a check in this box to enable gratuitous ARP (GARP)-based detection. Enabling ARP flooding enables GARP-based detection by default. This feature allows endpoints to be updated within this bridge domain when a GARP is received. A GARP is an ARP broadcast-type of packet that is used to verify that no other device on the network has the same IP address as the sending device. This check box appears only if you put a check in the **ARP Flooding** check box.
Unicast Routing check box	Put a check in this check box to enable unicast routing. Unicast routing is the forwarding method based on predefined forwarding criteria (IP or MAC address). This check box is unchecked by default. This check box appears only if you chose **Custom** in the **Forwarding** drop-down list.
IGMP Snoop Policy drop-down list	The Internet Group Management Protocol (IGMP) snooping policy. You do not need to choose an IGMP snooping policy to deploy Layer 4 to Layer 7 services.
Config BD MAC Address check box	Put a check in this check box to configure the bridge domain MAC address.

Name	Description
MAC Address field	The MAC address of the bridge domain. This field appears only if you put a check in the **Config BD MAC Address** check box.
Subnets section	The subnets of the bridge domain. Click +, complete the fields, and click **UPDATE** to add a subnet. You can add multiple subnets.
DHCP Labels section	The DHCP labels of the bridge domain. You do not need to configure a DHCP label to deploy Layer 4 to Layer 7 services. For information about configuring a DHCP label, see the *Cisco APIC Getting Started Guide*.

Step 2 Click **OK**. The next **NETWORK** page appears, and the bridge domain is created. On this page, you can add the Layer 2 (L2) external cache, the L3 external cache, additional networks, and additional bridge domains.

Step 3 Click **Next**. The **APPLICATION** page appears, which is used to configure application profiles. To configure an application profile, see #unique_36.

Configuring a Layer 3 Network

You can configure a Layer 3 (L3) network.

Before You Begin

- Configure a tenant. See #unique_29.

Step 1 On the **NETWORK** page of the **CREATE TENANT** dialog box, click + to add a network. The **CREATE NEW NETWORK** dialog box appears.

Step 2 Complete the following fields:

Name	Description
Name field	The name of the network.
Description field	The description of the network.
BGP Timers drop-down list	Choose the Border Gateway Protocol (BGP) timers of the network. To configure a new BGP timers policy, see Configuring a BGP Timers Policy.

Name	Description
OSPF Timers drop-down list	The Open Shortest Path First (OSPF) timers policy. The OSPF timer policy provides the Hello timer and Dead timer intervals configuration. You can choose the default policy or create a new policy. You do not need to choose a OSPF timer policy to deploy Layer 4 to Layer 7 services.
Monitoring Policy drop-down list	Choose the monitoring policy of the network.

Step 3 Click **Next**. The **BRIDGE DOMAIN** page appears, and the L3 network is created. To configure the bridge domain, see #unique_38.

Configuring an Application Profile

You can configure an application profile.

Before You Begin

- Configure a bridge domain. See #unique_38.

Step 1 On the menu bar, click the **TENANTS** tab. The **Tenant** window appears.

Step 2 On the submenu bar, click the tab of the tenant for which you want to configure an application profile. The **Tenant** window for the selected tenant appears in the **Work** pane.

Step 3 In the **Navigation** pane, expand the tenant's branch.

Step 4 Click **Application Profiles**. The **Application Profiles** window appears in the **Work** pane.

Step 5 Choose **ACTIONS** > **Create Application Profile**. The **CREATE APPLICATION PROFILE** dialog box appears.

Step 6 Complete the following fields:

Name	Description
Name field	The name of the application profile.
Alias field	The alias for the application profile. The alias can be a simpler and more descriptive name than the application profile's name when referring to a single application profile. You can assign a particular alias name to only one application profile; the system will prevent you from assigning the same alias name to a second application profile.
Description field	The priority of the application profile.
Tags field	A search keyword or term that is assigned to the application profile. A tag allows you to group multiple objects by a descriptive name. You can assign the same tag name to multiple objects and you can assign one or more tag names to an object.

Name	Description
Monitoring Policy drop-down list	Choose the endpoint group (EPG) monitoring policy name.

Step 7 In the **EPGs** section, click +. The **CREATE APPLICATION EPG** dialog box appears.

Step 8 Complete the following fields:

Name	Description
Name field	The name of the application EPG.
Alias field	The alias for the application EPG. The alias can be a simpler and more descriptive name than the application EPG's name when referring to a single application EPG. You can assign a particular alias name to only one application EPG; the system will prevent you from assigning the same alias name to a second application EPG.
Description field	The description of the application EPG.
Tags field	A search keyword or term that is assigned to the application EPG. A tag allows you to group multiple objects by a descriptive name. You can assign the same tag name to multiple objects and you can assign one or more tag names to an object.
QoS class drop-down list	Choose the quality of service priority class ID.
Custom QoS drop-down list	The quality of service traffic priority class ID. The custom class is a user-configurable differentiated services code point (DSCP) value. You do not need to choose a quality of service traffic priority class ID to deploy Layer 4 to Layer 7 services.
Bridge Domain drop-down list	Choose the name of the bridge domain that is associated with the application EPG.
Monitoring Policy drop-down list	Choose the endpoint group (EPG) monitoring policy name.
Associated Domain Profiles (VMs or bare metals) section	The domain profiles that are associated with the application EPG. Click + to add a domain profile. You can add more than one domain profile.
Statically Link with Leaves/Paths check box	Check this check box to link the application EPG statically with leafs and paths.

Step 9 If you checked the **Statically Link with Leaves/Paths** check box, click **NEXT**. The **LEAVES/PATHS** page appears.

a) In the **Leaves** section, click + to add a leaf.

b) Complete the following fields:

Name	Description
Node drop-down list	Choose the node to use as a leaf.
Encap field	The VLAN to use for encapsulation. The range is from 1 to 4094.

Name	Description
Deployment Immediacy drop-down list	Choose whether the deployment of this leaf association will occur immediately or when needed.
Mode drop-down list	Choose the mode of the static association with the leaf.

c) Click **UPDATE**. The leaf is added.
d) In the **Paths** section, click + to add a path.
e) Complete the following fields:

Name	Description
Path drop-down list	Choose the node to use as a path.
Encap field	The VLAN to use for encapsulation. The range is from 1 to 4094.
Deployment Immediacy drop-down list	Choose whether the deployment of this path association will occur immediately or when needed.
Mode drop-down list	Choose the mode of the static association with the path.

f) Click **UPDATE**. The path is added.

Step 10 Click **OK**. The **CREATE APPLICATION EPG** dialog box closes.

Step 11 In the **Provided Contracts** section, click + to add a provided contract. The **ADD PROVIDED CONTRACT** dialog box appears.

Step 12 Complete the following fields:

Name	Description
Contract Type drop-down list	Choose the type of the contract.
Name drop-down list	Choose the name of the contract. You can select **default**, a preexisting contract, or **Create New Contract**.

Step 13 Click **OK**. The **ADD PROVIDED CONTRACT** dialog box closes.

Step 14 n the **Consumed Contracts** section, click + to add a provided contract. The **ADD CONSUMED CONTRACT** dialog box appears.

Step 15 Complete the following fields:

Name	Description
Contract Type drop-down list	Choose the type of the contract.

Name	Description
Name drop-down list	Choose the name of the contract. You can select **default**, a preexisting contract, or **Create New Contract**.

Step 16 Click **OK**. The **ADD CONSUMED CONTRACT** dialog box closes.

Step 17 If any neighbors exist, in the **Neighbors** section, click + to add a neighbor. The **ADD NEIGHBOR** dialog box appears.

Step 18 Click **SUBMIT**. The **CREATE APPLICATION PROFILE** dialog box closes, and the application profile is configured.

Configuring a Contract

You can configure a contract.

Before You Begin

- Configure a tenant. See #unique_29.
- Configure a device cluster on the tenant. See Configuring a Device Cluster.

Step 1 On the menu bar, click the **TENANTS** tab. The **Tenant** window appears.

Step 2 In the **Navigation** pane, expand the tenant's tree and an application profile's tree under that tenant for which you want to configure a contract.

Step 3 Choose **Contracts**.

Step 4 Choose **Actions** > **Create Contract**. The **CREATE CONTRACT** dialog box appears.

Step 5 Complete the following fields:

Name	Description
Name field	The name of the contract.
Scope field	The scope of the contract.
Priority field	The priority of the contract.
Description field	The description of the contract.

Step 6 In the **Subjects** section, click + to add a contract subject. The **CREATE CONTRACT SUBJECT** dialog box appears.

Step 7 Complete the following fields:

Name	Description
Name field	The name of the contract subject.

Name	Description
Description field	The description of the contract subject.

Step 8 In the **Filter Chain** section, click + to add a filter.

Step 9 Choose the tenant for which the filter applies, and choose a service graph to use with the filter. Any traffic that is matched by the contract is redirected to the service graph.

Step 10 Click **UPDATE**. The filter is created.

Step 11 Click **OK**. The **CREATE CONTRACT SUBJECT** dialog box closes, and the contract subject is created.

Step 12 Click **SUBMIT**. The **CREATE CONTRACT** dialog box closes, and the contract is created.

Configuring a Management Endpoint Group

You can configure a new management endpoint group (EPG) to use with a device cluster.

Before You Begin

- Configure a tenant. See #unique_29.
- Configure a device cluster on the tenant. See Configuring a Device Cluster.

Step 1 From the **Create Device Cluster** dialog box, choose **Create Management EPG** from the **EPG** drop-down list. The **Create Management EPG** dialog box appears.

Step 2 Complete the following fields:

Name	Description
Application Profile field	The application profile to use with the management EPG. If a profile does not already exist, you can create one by choosing **Create Application Profile**.
Name field	The name for the application profile.
Alias field	The alias for the management EPG. The alias can be a simpler and more descriptive name than the management EPG's name when referring to a single management EPG. You can assign a particular alias name to only one management EPG; the system will prevent you from assigning the same alias name to a second management EPG.
Description field	The description of the management EPG.
QoS class drop-down list	Choose the quality of service class of the management EPG. If your device cluster uses physical appliances, choose **Unspecified**.

Name	Description
Bridge Domain drop-down list	Choose the bridge domain node to use with the management EPG.

Step 3 In the **Domains (VMM, Physical, or External) Associated to Interfaces** section, click + to add a domain.

Step 4 Complete the following fields:

Name	Description
Domain Profile drop-down list	Choose the domain profile to use with this management EPG.
Deployment Immediacy drop-down list	Choose whether to deploy the domain immediately or when needed.
Resolution Immediacy drop-down list	Choose how policies are pushed to leaf nodes: • **Immediate—**All policies, including VLAN bindings, NVGRE bindings, VXLAN bindings, contracts, and filters, are pushed to leaf nodes upon attaching a Hypervisor physical NIC. The Link Layer Discovery Protocol (LLDP) or OpFlex used to resolve Hypervisor-to-leaf node attachment. • **On-Demand—**Policies are pushed to leaf nodes only upon attaching a physical NIC and associating a virtual NIC with a port group (an EPG).

Step 5 In the **Reserved IP addresses for APICs** section, click + to create an IP address pool. The **CREATE IP ADDRESS POOL** dialog box appears.

Step 6 Complete the following fields:

Name	Description
Name field	The name for the IP address pool.
Gateway Address fields	The gateway's IP address and subnet mask.

Step 7 In the **Address Ranges** section, click +.

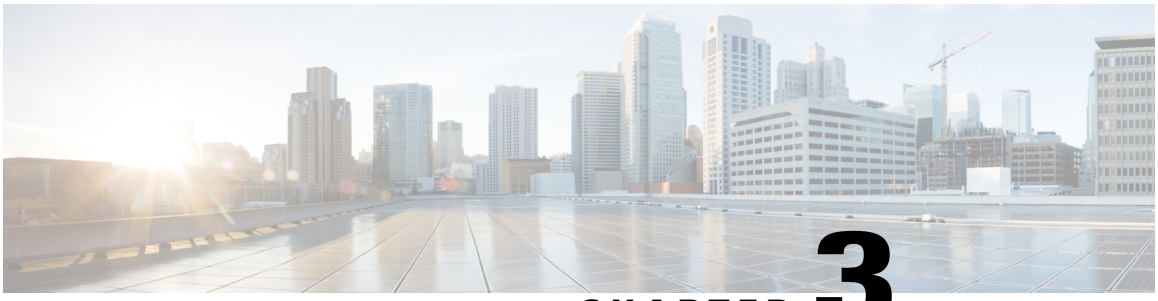
Step 8 Complete the following fields:

Name	Description
From field	The starting point of the IP address range.
To field	The ending point of the IP address range.

Step 9 Click **UPDATE**. The address range gets added to the **Address Ranges** section.

Step 10 Click **OK**. The **CREATE IP ADDRESS POOL** dialog box closes.

Step 11 Click **Submit**. The **Create Management EPG** dialog box closes and the management EPG is created.



CHAPTER 3

Importing a Device Package

Importing a Device Package

About the Device Package

The Application Policy Infrastructure Controller (APIC) requires a device package to configure and monitor service devices. A device package manages a class of service devices and provides the APIC with information about the devices so that the APIC knows what the device is and what the device can do. A device package is a `zip` file that contains the following parts:

Device specification	An XML file that defines the following properties: • Device properties: ◦ **Model—**Model of the device. ◦ **Vendor—**Vendor of the device. ◦ **Version—**Software version of the device. • Functions provided by a device, such as load balancing, content switching, and SSL termination. • Interfaces and network connectivity information for each function. • Device configuration parameters. • Configuration parameters for each function.
Device script	A Python script that performs the integration between the APIC and a device. The APIC events are mapped to function calls that are defined in the device script.

Function profile	A profile of parameters with default values that are specified by the vendor. You can configure a function to use these default values.
Device-level configuration parameters	A configuration file that specifies parameters that are required by a device at the device level. The configuration can be shared by one or more of the graphs that are using the device.

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

Example of Installing a Device Package

When you create a device package, you need to install it on the system.

The following is an example of installing a device package:

`$ scp AcmeADC.zip admin@10.10.10.10:`	Upload the package to the system.
`$ ssh admin@10.10.10.10`	Log in as a provider administrator.
`admin@apic:~> services install AcmeADC.zip`	Install the device package.
`admin@apic:~> rm AcmeADC.zip`	(Optional) Clean up the uploaded file. **Note** This will not uninstall the package.

Notes for Installing a Device Package with REST

- A device package can be installed using an HTTP or HTTPS POST.
- If HTTP is enabled on APIC, the URL for the POST is "http://10.10.10.10/ppi/node/mo/.xml".
- If HTTPS is enabled on APIC, the URL for the POST is "https://10.10.10.10/ppi/node/mo/.xml".
- The message must have a valid session cookie.

  See the *Cisco APIC REST API User Guide* for information about obtaining a 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

You can import a device package into the Application Policy Infrastructure Controller (APIC) so that the APIC knows what devices you have and what the devices can do.

Step 1 Log in as the provider administrator.

Step 2 On the menu bar, click the **L4-L7 Services** tab. The **Quick Start** window of the **INVENTORY** submenu tab appears.

Step 3 On the submenu bar, click the **PACKAGES** tab. The **Quick Start** window of the **PACKAGES** submenu tab appears.

Step 4 In the **Navigation** pane, click **Device Types**. The **Device Types** window appears.

Step 5 Choose **Actions** > **Import Device Package**. The **IMPORT DEVICE PACKAGE** dialog box appears.

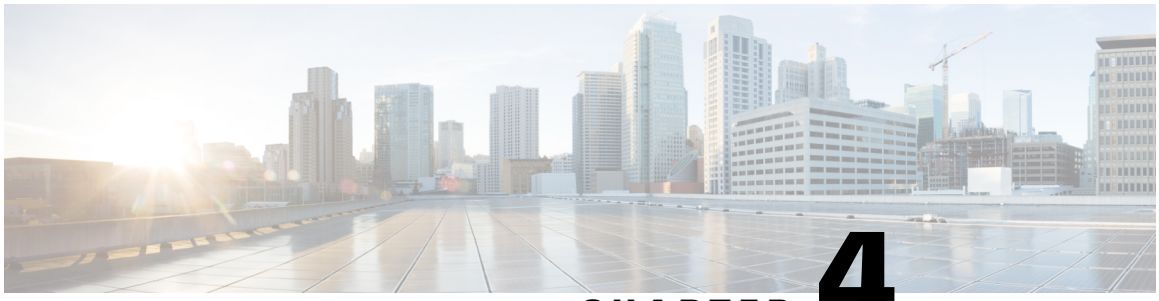
Step 6 In the **File Name** field, specify a device package that was either provided by the vendor or that you had previously created. For information about creating device packages, see *Cisco APIC Layer 4 to Layer 7 Device Package Development Guide* . As an example, the file `AcmeADC.zip`, which contains the device package, is selected and uploaded.

Step 7 Click **Submit**. The **IMPORT DEVICE PACKAGE** dialog box closes. A confirmation message appears if your package was successfully uploaded.

Step 8 Refresh the **Device Types** window in the APIC. The new device appears in the list of device types.

Step 9 (Optional) In the **Navigation** pane, expand **Device Types** to see the function parameters for the device package, and choose one of the functions.

Importing a Device Package



CHAPTER 4

Configuring a Device Cluster (Logical Device)

Configuring a Device Cluster (Logical Device)

About Device Clusters (Logical Devices)

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 logical interfaces, which describe the interface information for the device cluster. During service graph rendering, function node connectors are associated with 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 instantiation and rendering and programs the network resources onto the logical interfaces.

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

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

About Concrete Devices

A concrete device has concrete interfaces. When a concrete device is added to a logical device cluster, concrete interfaces are mapped to the logical interfaces. During service graph instantiation, VLANs and VXLANs are programmed on concrete interfaces that are based on their association with logical interfaces.

Configuring a Device Cluster

You can configure a device cluster.

Before You Begin

- Configure a tenant. See #unique_50.

Step 1 On the menu bar, click the **TENANTS** tab. The **Tenant** window appears.

Step 2 In the **Navigation** pane, expand the **Tenant** branch, expand the **L4-L7 Services** branch, and click **Device Clusters**. The **Device Clusters** window appears.

Step 3 Choose **Actions** > **Create Device Cluster**. The **CREATE DEVICE CLUSTER** dialog box appears, showing the **CLUSTER** page.

Step 4 Complete the following fields:

Name	Description
Name field	The name of the device cluster.
Device Package field	The device package that you uploaded.
Context Aware radio buttons	You can allocate a device to a tenant. **Single** means that the device cluster cannot be shared across multiple tenants of a given type that are hosted on the provider network, and you must give the device cluster to a specific tenant for a given user. **Multiple** means that the device cluster can be shared across multiple tenants of a given type that you are hosting on this provider network. For example, there could be two hosting companies that share the same device. The tenancy assignment is implicitly based on the endpoint group (EPG) to which the package is bound. If you created a cluster, you must specify the management EPG, which determines the network through which the device would be managed.
Function Type radio buttons	A **GoThrough** device is a transparent device. A packet goes through without being addressed to the device, and endpoints are not aware of that device. A **GoTo** device has a specific destination, depending on the package. In this example, the device package indicates that it only supports **GoTo** functions, so the radio buttons are grayed out and **GoTo** is automatically selected. If the device package supports both function types, you can select one of the radio buttons to indicate the type of function and how you will configure it. **GoTo** is the default value.
Device Type radio buttons	Specifies whether the device cluster has **PHYSICAL** appliances or **VIRTUAL** appliances.
Physical Domain drop-down list	Choose the domain to use when allocating network resources for the graphs that will use this device cluster. You only specify the physical domain for a physical device type. You can select an existing physical domain or configure a new one. To configure a new physical domain, see #unique_51.
VMM Domain drop-down list	Choose the Virtual Machine Manager (VMM) domain, which is also known as a vCenter domain. You only specify the VMM domain for a virtual device type. You can select an existing VMM domain or configure a new one. To configure a new VMM domain, see #unique_52.

Name	Description
EPG drop-down list	Choose the management EPG to indicate which management network leads to the device that you are trying to manage. You do not need to choose a management EPG if the device is managed in-band. To configure a new management EPG, see Configuring a Management Endpoint Group, on page 15. You do not need to choose a management EPG if the management happens out of band from an external network.
Virtual IP Address field	The virtual IP address of the management interface for the concrete device in the device cluster.
Port field	The port number of the management interface for the concrete device in the device cluster.
Username field	The username for logging into the device cluster.
Password field	The password for logging into the device cluster.
Confirm Password field	The confirmation of the password for logging into the device cluster.

Step 5 In the **Logical Interfaces** section, click + to add a logical interface. When a device cluster is used by a graph, the graph's connectors are mapped to a logical interface.

Step 6 Complete the following fields:

Name	Description
Name field	The name of the logical interface.
Type field and drop-down list	The type of logical interface. You must choose or enter the type that is specified in the device package.

Step 7 Click **Next**. The **DEVICES** page appears.

Step 8 Add concrete devices and concrete interfaces. See Configuring a Concrete Device, on page 24.

Step 9 After adding concrete interfaces, click **Next**. The **PARAMETERS** page appears.

Step 10 (Optional) Add configuration parameters. The configuration parameters are for the concrete device, and are used during the one-time configuration at initialization time.

Step 11 Click **OK**. The **CREATE CONCRETE DEVICE** dialog box closes, taking you back to the **DEVICES** page of the **CREATE DEVICE CLUSTER** dialog box.

Step 12 Click **Finish** to create the device. The **CREATE DEVICE CLUSTER** dialog box closes, taking you back to the **Tenant** page.

Step 13 On the submenu bar, click the **common** tab and refresh the page to see the device. If the device is down or has been incorrectly configured, it will not appear. In an incorrect configuration, the APIC might not have been able to validate the device's presence, the device is present but is not the right kind, or the device is running a version that cannot be verified through the device package. In such cases, device verification fails, and you should be able to see the fault indicating the reason for device registration failure.

Configuring a Concrete Device

You can configure a concrete device.

Before You Begin

- Configure a tenant. See #unique_50.
- Configure a device cluster on the tenant. See Configuring a Device Cluster, on page 21.

Step 1 On the **DEVICES** page of the **CREATE DEVICE CLUSTER** dialog box, click + in the **Create L4-L7 Devices** section to add a concrete device to the device cluster. The **CREATE CONCRETE DEVICE** dialog box appears.

Step 2 Complete the following fields:

Name	Description
Name field	The name of the concrete device.
VM Name field	For virtual device types only. The name of the virtual machine on which the device is hosted.
vCenter Name field	For virtual device types only. The name of the VMM domain on which the device is hosted.
IP Address field	The management interface host address for the concrete device in the device cluster.
Port field	The management interface port number for the concrete device in the device cluster.
Username field	The username for logging into the device cluster.
Password field	The password for logging into the device cluster.
Confirm Password field	The confirmation of the password for logging into the device cluster.

Step 3 Click + to add a concrete interface, which is the interface on the concrete device, and complete the following fields:.

Name	Description
Name field	The name field identifies an interface on the concrete device. For example, 1/0, 1.1, or Gig 0/0. **Note** Certain device packages require '/' in their interface name to be replaced with '_'; that is interface 0/1 on the device should be represented as 0_1 on the APIC. Refer to the device package documentation to see whether the device can support '/' in its naming property or if the '/' should be replaced with '_'.
Path field	For physical devices. Specifies how the concrete interface attaches to the fabric. For example, the leaf node/slot/port to which the concrete interface is attached.

Name	Description
vNIC field	The network adapter name that was assigned on the vCenter for identifying the corresponding interface of a concrete device. Usually a vNIC is labeled **Network adapter x** on the vCenter where x = 1, 2, 3 … **Note** You can check the interface MAC on the device and then identify the corresponding vNIC on the vCenter by matching the MAC field.
Logical Interface field	The type of logical interface to which the concrete interface is mapped.

The information that you enter specifies how the concrete interfaces are connected to the fabric and how the concrete interfaces are mapped to the logical interfaces.

Step 4 Click **Update** after entering the information for a concrete interface to add the interface to the concrete device.

Step 5 Click **Next**. The **PARAMETERS** page appears.

Step 6 (Optional) Add configuration parameters. The configuration parameters are for the concrete device and are used during the one-time configuration at initialization time.

Step 7 Click **OK**. The **CREATE CONCRETE DEVICE** dialog box closes, taking you back to the **DEVICES** page of the **CREATE DEVICE CLUSTER** dialog box, and the concrete device is configured.

Configuring a Device Cluster for a Virtual Service Device

You can configure a device cluster for a virtual service device.

Before You Begin

- Configure a tenant. See #unique_50.

Step 1 In the **Navigation** pane of the **Tenant common** window, expand the **Tenant common** branch, expand the **L4-L7 Services** branch, and click **Device Clusters**. The **Device Clusters** window appears.
Note A device cluster can also be created under the **User Tenant** space.

Step 2 Choose **Actions** > **Create Device Cluster**. The **CREATE DEVICE CLUSTER** dialog box appears.

Step 3 For the **Device Type**, choose **VIRTUAL**.

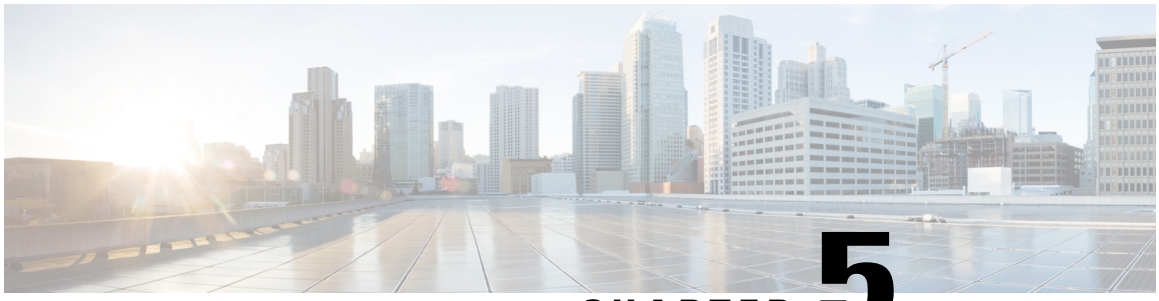
Step 4 For the **VMM Domain**, you must associate your Virtual Machine Manager (VMM) domain to the virtual logical device cluster. This VMM domain must have the information on how to connect to your vCenter, such as the IP address, port, and administrator's credentials. Also, you must specify a VLAN pool, which is used to allocate VLANs for the service graph.

Step 5 Complete the remainder of the required device cluster information. For more information about the device cluster information, see Configuring a Device Cluster, on page 21. Continue with that procedure until you must add concrete devices.

Step 6 When you add the concrete devices, for each concrete device, you must provide the vCenter name, the virtual machine name, the vNIC name, and the MAC address for each interface. For more information about adding concrete devices,

see Configuring a Concrete Device, on page 24. The Application Policy Infrastructure Controller (APIC) creates the appropriate port groups for the service appliance and places the interfaces into the correct port groups. You do not need to place the NICs.

Step 7 Continue with the procedure for configuring a device cluster. See Configuring a Device Cluster, on page 21.



CHAPTER 5

Configuring Connectivity to Device Cluster

About Configuring Connectivity to Device Cluster

APIC needs to connect to service appliances to be able to configure them. This requires connectivity between APIC and the service appliances to be configured. APIC connects to the management interface of the service appliance using the management IP configured on service appliance.

You should configure the management interface and assign the management IP to the service appliance before connecting it to the ACI fabric. Then you configure the in-band connectivity between APIC and service appliance such that APIC can connect to the management IP of the service appliance.

APIC supports two ways to connect to management IP of service appliance:

- In-band connectivity to devices in device clusters using the tenant's VRF.
- In-band connectivity to devices in device clusters using the management tenant VRF.

About In-band Connectivity to Devices in Device Clusters Using Tenant's VRF

In this scenario, the tenant administrator owns and manages the IP addresses assigned to the service appliance management interface (management IP of the service appliance) and the APIC. In this scenario, service appliance's management IP is in the tenant's VRF and APIC connects to the management IP of the service appliance using a source IP that is also in tenant's VRF.

About In-band Connectivity to Devices in Device Clusters Using Management Tenant VRF

APIC provides a pre-configured tenant **tn-mgmt**. It comes with a pre-configured VRF. The management IP address assigned to a service appliance can be owned and managed by **tn-mgmt**. In this scenario, the service appliance's management IP is in the tn-mgmt VRF and APIC connects to the management IP of the service appliance using a source IP that is also in the tn-mgmt's VRF.

Configuring In-Band Connectivity to Devices in Device Clusters Using Tenant's VRF

Before You Begin

- Configure a device cluster.

  See #unique_58.
- Configure a bridge domain.

  See #unique_38.
- Configure an endpoint group. (EPG)

  See Configuring a Management Endpoint Group, on page 15.

DETAILED STEPS

<table>
<tr><th></th><th>Command or Action</th><th>Purpose</th></tr>
<tr><td>Step 1</td><td>On the BRIDGE DOMAIN page of the CREATE TENANT dialog box, complete the following fields:
<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td>ARP Flooding</td><td>Disable ARP Flood by clearing the ARP Flood checkbox.</td></tr>
<tr><td>Subnets</td><td>Click +, complete the fields, and click UPDATE to add a subnet for to add a subnet for</td></tr>
</table></td><td></td></tr>
<tr><td>Step 2</td><td>Click OK. The next~NETWORK~ page appears, and the bridge domain is updated.</td><td></td></tr>
<tr><td>Step 3</td><td>From the Create Device Cluster dialog box, choose Create Management EPG from the EPG dropdown list. The Create Management EPG dialog box appears.</td><td></td></tr>
<tr><td>Step 4</td><td>Complete the following fields:
<table>
<tr><th>Option</th><th>Description</th></tr>
<tr><td>Bridge Domain drop-down list</td><td>Choose the bridge domain node to use with the management EPG.</td></tr>
</table></td><td></td></tr>
</table>

	Command or Action	Purpose
Step 5	Allocate the Controller Management IP addresses from the allocated subnet.	
Step 6	On the menu bar, click the **TENANTS** tab. The **TENANT** window appears.	
Step 7	On the submenu bar, click **ADD TENANT**. The **CREATE TENANT** dialog box appears, showing the **TENANT** page.	
Step 8	Complete the following fields: **Option** / **Description** **Controller Management Policies** section — Click + and complete the fields to provide each context with a separate controller management policy.	
Step 9	Point the controller management policies to the EPG.	
Step 10	Point the EPG to the device cluster.	

Using REST APIs to Configure In-Band Connectivity to Devices in Device Clusters Using Tenant's VRF

The following is an example of using REST APIs to configure in-band connectivity to devices in device clusters using tenant's VRF:

1 Define the EPG that is to be used for management.

Note Ensure to open up the ports to the domain mappings using the appropriate selectors configuration.

In the following, the EPG "services" is used for the management of the Services Devices/VMs subnet that is used for Tenant vrf devicemanagement (3.3.3.0/24).

```
<polUni>
  <fvTenant name="coke">
    <fvCtx name="mgmt_ctx1"/>
    <vnsCtrlrMgmtPol ctxDn="uni/tn-coke/ctx-mgmt_ctx1">
      <vnsRsMgmtAddr tDn="uni/tn-coke/ap-services/epg-ifc/CtrlrAddrInst-ifc"/>
    </vnsCtrlrMgmtPol>
    <fvBD name="mgmt_ServicesMgmtBD">
      <fvRsCtx tnFvCtxName="mgmt_ctx1"/>
      <fvSubnet ip="3.3.3.3/24"/>
    </fvBD>
    <fvAp name="services">
      <fvAEPg name="ifc">
        <fvRsBd tnFvBDName="mgmt_ServicesMgmtBD"/>
        <vnsAddrInst name="ifc">
          <fvnsUcastAddrBlk from="3.3.3.100/24" to="3.3.3.200/24"/>
        </vnsAddrInst>
        <fvRsDomAtt tDn="uni/vmmp-VMware/dom-mininet"/>
      </fvAEPg>
    </fvAp>
  </fvTenant>
```

```
</polUni>
```

2 Associate the EPG to the LDevVip.

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

        <vnsLDevVip name="ADCCluster1"
                    funcType="GoTo" devtype="VIRTUAL">
            <vnsRsMDevAtt tDn="uni/infra/mDev-Citrix-NetScaler-10.5"/>
            <vnsRsALDevToDomP tDn="uni/vmmp-VMware/dom-mininet"/>
            <vnsRsDevEpg tDn="uni/tn-coke/ap-services/epg-ifc"/>

            <vnsCMgmt name="devMgmt"
                      host="3.3.3.180"
                      port="80"/>

            <vnsCCred name="username"
                      value="nsroot"/>

            <vnsCCredSecret name="password"
                            value="nsroot"/>
        </vnsLDevVip>
    </fvTenant>
</polUni>
```

Configuring In-Band Connectivity to Devices in Device Clusters Using Management Tenant VRF

Step 1 In tn_mgmt, set up the in-band VRF connectivity from APIC to external nodes, such as VMware vCenter.
For more information, see the Configuring In-Band Management Access Using the GUI section in the *Cisco APIC Getting Started Guide*.

Step 2 In tn_mgmt, create a separate endpoint group (EPG), such as mgmt-services-epg, with a subnet for the service nodes. Use a contract to stitch the APICs in the in-band VRF.
For more information about EPGs, see Configuring a Management Endpoint Group, on page 15.

Step 3 In the device cluster screen, select the EPG, such as mgmt-services-epg.

Using REST APIs to Configure In-Band Connectivity to Devices in Device Clusters Using Management Tenant VRF

The following is an example of using REST APIs to configure in-band connectivity to devices in device clusters using management tenant VRF:

1 Create an EPG l4l7MgmtEpg in tenant management.

Note l4l7MgmtEpg is a part of bd access which is under inb context in tn-mgmt.

contract1 is the contract between the tn-mgmt l4l7MgmtEpg and tn-mgmt inb default EPG.

```
<polUni>
    <fvTenant dn="uni/tn-mgmt">
        <fvAp name="services">
            <fvAEPg name="l4l7MgmtEpg">
              <fvRsBd tnFvBDName="access" />
                <fvRsDomAtt tDn="uni/vmmp-VMware/dom-mininet" />
             <fvRsCons tnVzBrCPName='contract1'>
              </fvRsCons>
            </fvAEPg>
        </fvAp>
        <fvBD name="access">
            <fvSubnet ip="3.3.3.3/24" />
            <fvRsCtx tnFvCtxName="inb"/>
        </fvBD>
        <vzFilter name='all'>
            <vzEntry name='all' ></vzEntry>
        </vzFilter>
        <vzBrCP name="contract1" scope="tenant">
            <vzSubj name='subj1'>
                <vzInTerm>
                    <vzRsFiltAtt tnVzFilterName="all" />
                </vzInTerm>
                <vzOutTerm>
                    <vzRsFiltAtt tnVzFilterName="all" />
                </vzOutTerm>
            </vzSubj>
        </vzBrCP>
    </fvTenant>
</polUni>
```

2 Ensure that the Service Device/VM has the mgmt IP address in the subnet 3.3.3.0/24.

This is the same subnet that tn-mgmt access BD has been configured with. (See configuration in earlier step.)

3 Add the following to the LDevVip:

Note This points to the EPG that was created in the earlier step

<vnsRsDevEpg tDn="uni/tn-mgmt/ap-services/epg-l4l7MgmtEpg"/>.

```
<polUni>
    <fvTenant name="mgmt">

        <vnsLDevVip name="ADCCluster1"
                    funcType="GoTo" devtype="VIRTUAL">
            <vnsRsMDevAtt tDn="uni/infra/mDev-Citrix-NetScaler-10.5"/>
            <vnsRsALDevToDomP tDn="uni/vmmp-VMware/dom-mininet"/>
            <vnsRsDevEpg tDn="uni/tn-mgmt/ap-services/epg-l4l7MgmtEpg"/>

            <vnsCMgmt name="devMgmt"
                      host="3.3.3.180"
                      port="80"/>

            <vnsCCred name="username"
                      value="nsroot"/>

            <vnsCCredSecret name="password"
```

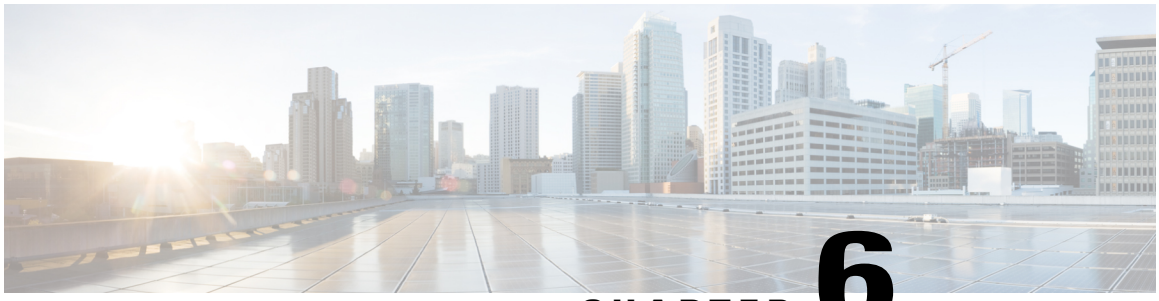
```
                                value="nsroot"/>

        </vnsLDevVip>

    </fvTenant>
</polUni>
```

4 Add the route in service Device/VM to point to the IFC inband gateway.

For example, on the route on netScaler, add route 3.0.0.0 255.255.255.0 3.3.3.3, where 3.0.0.0/24 is the IFC inband subnet and 3.3.3.3 is the SVI IP for l4l7MgmtEpg.

5 Verify the following:

- The route table on IFC has an entry for ifc inband IP.
- The IFC can ping the l4l7MgmtEpg gateway on the leaf.
- The service node can ping the l4l7MgmtEpg SVI gateway and IFC inb SVI Ip.



CHAPTER 6

Using a Device Cluster

Using a Device Cluster

About Device Cluster (Logical Device) Contexts

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

A device cluster context specifies the policy for selecting a device cluster for a service graph. This allows an administrator to have multiple device clusters and then be able to use them for different service graphs. For example, an administrator can have a device cluster that has high performance ADC appliances and another device cluster that has lower performance ADC appliances. Using two different device cluster contexts, one for the high performance ADC cluster and the other for the low performance ADC cluster, the administrator can select the high performance ADC cluster for the applications that require higher performance and select the low performance ADC cluster for the applications that require lower performance.

Configuring a Logical Device Context

You can configure a logical device context.

Before You Begin

- Configure a tenant. See #unique_29.

• Configure a device cluster on the tenant. See Configuring a Device Cluster.

Step 1 On the menu bar, click the **TENANTS** tab. The **Tenant** window appears.

Step 2 On the submenu bar, click the name of the tenant for which you want to configure a logical device context.

Step 3 In the **Navigation** pane, expand the tenant's branch.

Step 4 Expand the **L4-L7 Services** branch.

Step 5 Expand the **Device Cluster Selection Policies** branch. The **Device Cluster Selection Policies** window appears in the **Work** pane.

Step 6 Choose **ACTIONS** > **Create Logical Device Context**. The **CREATE LOGICAL DEVICE CONTEXT** dialog box appears.

Step 7 Complete the following fields:

Name	Description
Contract Name field	The name of the contract for the logical device context. If you do not want to use the contract name as part of the criteria for using a device cluster, choose **any**.
Graph Name field	The name of the graph for the logical device context. If you do not want to use the graph name as part of the criteria for using a device cluster, choose **any**.
Node Name field	The name of the function node for the logical device context. If you do not want to use the node name as part of the criteria for using a device cluster, choose **any**.
Device Cluster field	The device cluster to be used for the logical device context.

Step 8 In the **Logical Interface Contexts** section, click + to add a logical interface contexts.

Step 9 Complete the following fields:

Name	Description
Connector Name field	The name of the connector in the service graph.
Logical Interface field	The logical interface to use for the connector that is specified in the logical interface context.
Subnets field	The subnet to configure on the logical interfaces when the service graph is instantiated.

Step 10 Click **SUBMIT**. The **CREATE LOGICAL DEVICE CONTEXT** dialog box closes.

Configuring Using REST APIs

Configuring Using REST APIs to Create Device Cluster Context

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

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

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

             <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>
```

Configuring Using REST APIs to Add a Logical Interface in a Device Cluster

```
<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>
```

Configuring Using CLI Commands

```
# logical-device-context

cd '/aci/tenants/acme/l4-l7-services/device-cluster-selection-policies'
mocreate 'any' 'any' 'any'
cd 'any-any-any'
moset device-cluster 'tenants/acme/l4-l7-services/device-clusters/InsiemeCluster'
moconfig commit


# vns-lifctx

cd
'/aci/tenants/acme/l4-l7-services/device-cluster-selection-policies/webCtrt-G1-Node1/logical-interface-contexts'
mocreate 'int'
cd 'int'
moset logical-interface
'tenants/acme/l4-l7-services/device-clusters/InsiemeCluster/logical-interfaces/int'
moset bridge-domain 'tenants/acme/networking/bridge-domains/MySrvrBD'
moconfig commit


# vns-lifctx

cd
'/aci/tenants/acme/l4-l7-services/device-cluster-selection-policies/webCtrt-G1-Node1/logical-interface-contexts'
mocreate 'ext'
cd 'ext'
moset logical-interface
```

```
‘tenants/acme/l4-l7-services/device-clusters/InsiemeCluster/logical-interfaces/ext'
moset bridge-domain ‘tenants/acme/networking/bridge-domains/MyClntBD'
moconfig commit
```

Importing Device Clusters

If a cluster is defined in a common tenant (such as **tn-mgmt**), a tenant can import the configured device cluster as follows:

Step 1 On the menu bar, click the **L4-L7** Services tab.
Step 2 On the submenu bar, click the **PACKAGES** tab.
Step 3 In the **Navigation** pane, click **Device Types**.
Step 4 Choose **Actions** > **Import Device Package**. The **IMPORT DEVICE PACKAGE** dialog box appears.
Step 5 In the **File Name** field, specify a device package that was either provided by the vendor or that you had previously created.
Step 6 Click **Submit**. The **IMPORT DEVICE PACKAGE** dialog box closes.

Verifying Import of Device Clusters

Step 1 On the menu bar, click the **TENANTS** tab. The **Tenant** window appears.
Step 2 On the submenu bar, click the name of the tenant for which you want to import device clusters.
Step 3 In the **Navigation** pane, expand the tenant's branch.
Step 4 Expand the **L4-L7 Services** branch.
Step 5 Expand the **Imported Device Clusters** branch.
Step 6 Choose the appropriate device cluster. The device cluster information appears in the **Work** pane.

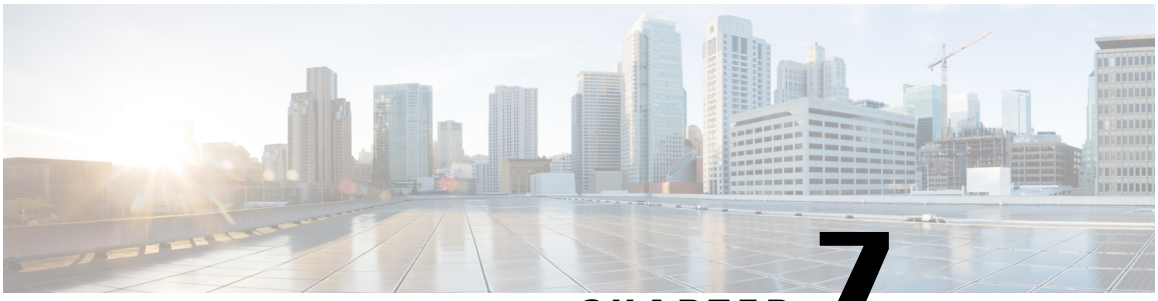
Importing Using REST APIs

```
<polUni>
  <fvTenant dn="uni/tn-coke" name="coke">
    <vnsLDevIf ldev="uni/tn-mgmt/lDevVip-ADCCluster1"/>
    <vnsLDevCtx ctrctNameOrLbl="any" graphNameOrLbl="any" nodeNameOrLbl="any">
      <vnsRsLDevCtxToLDev tDn="uni/tn-coke/lDevIf-[uni/tn-mgmt/lDevVip-ADCCluster1]"/>
      <vnsLIfCtx connNameOrLbl="inside">
        <vnsRsLIfCtxToLIf
tDn="uni/tn-coke/lDevIf-[uni/tn-mgmt/lDevVip-ADCCluster1]/lDevIfLIf-inside"/>
        <fvSubnet ip="10.10.10.10/24"/>
        <vnsRsLIfCtxToBD tDn="uni/tn-coke/BD-cokeBD1"/>
      </vnsLIfCtx>
      <vnsLIfCtx connNameOrLbl="outside">
        <vnsRsLIfCtxToLIf
```

```
tDn="uni/tn-coke/lDevIf-[uni/tn-mgmt/lDevVip-ADCCluster1]/lDevIfLIf-outside"/>
        <fvSubnet ip="70.70.70.70/24"/>
        <vnsRsLIfCtxToBD tDn="uni/tn-coke/BD-cokeBD4"/>
      </vnsLIfCtx>
    </vnsLDevCtx>
  </fvTenant>
</polUni>
```

Importing Using CLI Commands

```
# logical-device-context
cd '/aci/tenants/coke/l4-l7-services/device-cluster-selection-policies'
mocreate 'any' 'any' 'any'
cd 'any-any-any'
moset device-cluster
'tenants/coke/l4-l7-services/imported-device-clusters/[uni/tn-mgmt/lDevVip-ADCCluster1]'
moconfig commit


# vns-lifctx
cd
'/aci/tenants/coke/l4-l7-services/device-cluster-selection-policies/any-any-any/logical-interface-contexts'
mocreate 'inside'

cd 'inside'

moset logical-interface
'uni/tn-coke/lDevIf-[uni/tn-mgmt/lDevVip-ADCCluster1]/lDevIfLIf-inside'
moset bridge-domain 'tenants/coke/networking/bridge-domains/cokeBD1'

moconfig commit


# fv-subnet
cd
'/aci/tenants/coke/l4-l7-services/device-cluster-selection-policies/any-any-any/logical-interface-contexts/inside/subnets'
mocreate '10.10.10.10/24'

moconfig commit


# vns-lifctx
cd
'/aci/tenants/coke/l4-l7-services/device-cluster-selection-policies/any-any-any/logical-interface-contexts'
mocreate 'outside'

cd 'outside'

moset logical-interface
'uni/tn-coke/lDevIf-[uni/tn-mgmt/lDevVip-ADCCluster1]/lDevIfLIf-outside'
moset bridge-domain 'tenants/coke/networking/bridge-domains/cokeBD4'

moconfig commit


# fv-subnet
cd
'/aci/tenants/coke/l4-l7-services/device-cluster-selection-policies/any-any-any/logical-interface-contexts/outside/subnets'
mocreate '70.70.70.70/24'

moconfig commit


# logical-device-consumer
cd '/aci/tenants/coke/l4-l7-services/imported-device-clusters'
mocreate 'uni/tn-mgmt/lDevVip-ADCCluster1'
cd '[uni--tn-mgmt--lDevVip-ADCCluster1]'
moset faultcode '0'
moconfig commit
```



CHAPTER 7

Configuring a Service Graph

Configuring a Service Graph

About Service Graphs

A service graph is an ordered set of function nodes between a set of terminals, which identifies a set of network service functions that are required by an application. Service functions within a graph are automatically provisioned on a service device that is based on an application's requirements.

For more information about function nodes, see #unique_83.

You can define a service graph by using the GUI, CLI, or the Application Policy Infrastructure Controller (APIC). Configuring a service device through the APIC does not require changes on a service device.

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. For more information about function node connectors, see #unique_84.

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 context. 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.

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 Configuration Parameters

A service graph 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 can require one or more configuration parameters. The parameter values can be locked to prevent any additional changes.

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

Configuring a Service Graph

After you register a device, you can create service graphs using that device and all the functions that device has exposed. The service graph can be created under tenant common or can be tenant specific. This can be done by the provider administrator or by the tenant administrator within its own tenancy.

Step 1 In the **Navigation** pane, click **L4-L7 Services** > **Service Graphs**.

Step 2 Choose **Action** > **Create L4-L7 Service Graph**. The **CREATE L4-L7 SERVICE GRAPH** dialog box appears. The left pane lists the service devices that the Application Policy Infrastructure Controller (APIC) knows about and lists the service functions that are provided by the devices. The APIC knows about these devices from the device package that you imported.

Step 3 Complete the following fields:

Name	Description
Name field	The name of the service graph.

Name	Description
Description field	The description of the service graph.

Step 4 Drag and drop a service function from the left pane to the right pane to add that function to the service graph.

Step 5 Based on the device package, click on the function connectors, int and ext, to map them respectively to the meta connectors, Inside and Outside. the meta connectors, Inside and Outside, are device specific.

Step 6 Connect the terminal nodes to the connectors of the service function by clicking a terminal node and dragging a connection to a connector of the function. Repeat this action for each terminal node that you want to connect to the function. Terminal nodes represent the connectivity to the application EPG.

Step 7 Click the function. The **Config Profile** dialog box appears, prompting you to choose a profile for the service node. You can configure the service function parameters at this stage, or you can do so at the graph attachment time when a graph is associated with the contract.

Step 8 Once you have completed configuring the service function parameters, return to the **CREATE L4-L7 SERVICE GRAPH** dialog box and click **Submit** to create the graph. The **Service Graph** dialog box lists the new graph that you created.

Configuring Using REST APIs to Create a Service Graph

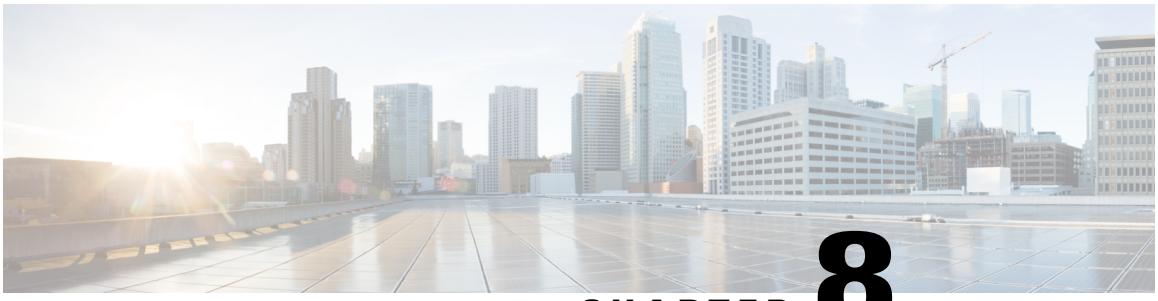
```
<polUni>
  <fvTenant name="acme">
    <vnsAbsGraph name="G1">
      <vnsAbsTermNodeCon name="Input1">
        <vnsAbsTermConn name="C1">
                </vnsAbsTermConn>
      </vnsAbsTermNodeCon>
      <vnsAbsNode name="Node" funcType="GoTo">
        <vnsRsDefaultScopeToTerm
tDn="uni/tn-acme/AbsGraph-G1/AbsTermNodeProv-Output1/outtmnl"/>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

```
      <vnsAbsConnection name="CON1">
        <vnsRsAbsConnectionConns
tDn="uni/tn-acme/AbsGraph-G1/AbsTermNodeCon-Input1/AbsTConn"/>
       <vnsRsAbsConnectionConns tDn="uni/tn-acme/AbsGraph-G1/AbsNode-Node/AbsFConn-inside"/>

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

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

Configuring Using CLI Commands

```
# l4-l7-service-graph

cd '/aci/tenants/asa_tenant1/l4-l7-services/service-graphs'
mocreate 'g1'
moconfig commit


# consumer-terminal-node

cd '/aci/tenants/asa_tenant1/l4-l7-services/service-graphs/g1/consumer-terminal-nodes'
mocreate 'Input1'
moconfig commit


# terminal-connector

cd '/aci/tenants/asa_tenant1/l4-l7-services/service-graphs/g1/consumer-terminal-nodes/Input1'
mocreate terminal-connector
cd 'terminal-connector'
moset name 'C1'
moconfig commit


# provider-terminal-node

cd '/aci/tenants/asa_tenant1/l4-l7-services/service-graphs/g1/provider-terminal-nodes'
mocreate 'Output1'
moconfig commit


# terminal-connector

cd '/aci/tenants/asa_tenant1/l4-l7-services/service-graphs/g1/provider-terminal-nodes/Output1'
mocreate terminal-connector
cd 'terminal-connector'
moset name 'C6'
moconfig commit


# function-node

cd '/aci/tenants/asa_tenant1/l4-l7-services/service-graphs/g1/function-nodes'
mocreate 'ASA_FW'
cd 'ASA_FW'
moset function-name 'l4-l7-services/packages/CISCO-ASA-1.0.1.37/functions/Firewall'
moconfig commit


# device-config

cd '/aci/tenants/asa_tenant1/l4-l7-services/service-graphs/g1/function-nodes/ASA_FW'
mocreate device-config
```

```
moconfig commit


# function-connector

cd '/aci/tenants/asa_tenant1/l4-l7-services/service-graphs/g1/function-nodes/ASA_FW'
mocreate function-connector 'external'
cd 'function-connector-external'
moset meta-connector 'uni/infra/mDev-CISCO-ASA-1.0.1.37/mFunc-Firewall/mConn-external'
moconfig commit


# function-connector

cd '/aci/tenants/asa_tenant1/l4-l7-services/service-graphs/g1/function-nodes/ASA_FW'
mocreate function-connector 'internal'
cd 'function-connector-internal'
moset meta-connector 'uni/infra/mDev-CISCO-ASA-1.0.1.37/mFunc-Firewall/mConn-internal'
moconfig commit


# default-scope-to-term

cd '/aci/tenants/asa_tenant1/l4-l7-services/service-graphs/g1/function-nodes/ASA_FW'
mocreate default-scope-to-term
cd 'default-scope-to-term'
moset target-dn 'uni/tn-asa_tenant1/AbsGraph-g1/AbsTermNodeProv-Output1/outtmnl'
moconfig commit


# connection

cd '/aci/tenants/asa_tenant1/l4-l7-services/service-graphs/g1/connections'
mocreate 'CON2'
moconfig commit


# relation-from-abstract-connection-to-abstract-connectors

cd '/aci/tenants/asa_tenant1/l4-l7-services/service-graphs/g1/connections/CON2'
mocreate
'tenants/asa_tenant1/l4-l7-services/service-graphs/g1/function-nodes/ASA_FW/function-connector-internal'
moconfig commit


# relation-from-abstract-connection-to-abstract-connectors

cd '/aci/tenants/asa_tenant1/l4-l7-services/service-graphs/g1/connections/CON2'
mocreate
'tenants/asa_tenant1/l4-l7-services/service-graphs/g1/provider-terminal-nodes/Output1/terminal-connector'
moconfig commit


# connection

cd '/aci/tenants/asa_tenant1/l4-l7-services/service-graphs/g1/connections'
mocreate 'CON1'
moconfig commit


# relation-from-abstract-connection-to-abstract-connectors

cd '/aci/tenants/asa_tenant1/l4-l7-services/service-graphs/g1/connections/CON1'
mocreate
'tenants/asa_tenant1/l4-l7-services/service-graphs/g1/consumer-terminal-nodes/Input1/terminal-connector'
moconfig commit


# relation-from-abstract-connection-to-abstract-connectors

cd '/aci/tenants/asa_tenant1/l4-l7-services/service-graphs/g1/connections/CON1'
mocreate
'tenants/asa_tenant1/l4-l7-services/service-graphs/g1/function-nodes/ASA_FW/function-connector-external'
moconfig commit
admin@ifav15-ifc2:g1>
```



CHAPTER 8

Configuration Parameters

Configuration Parameters

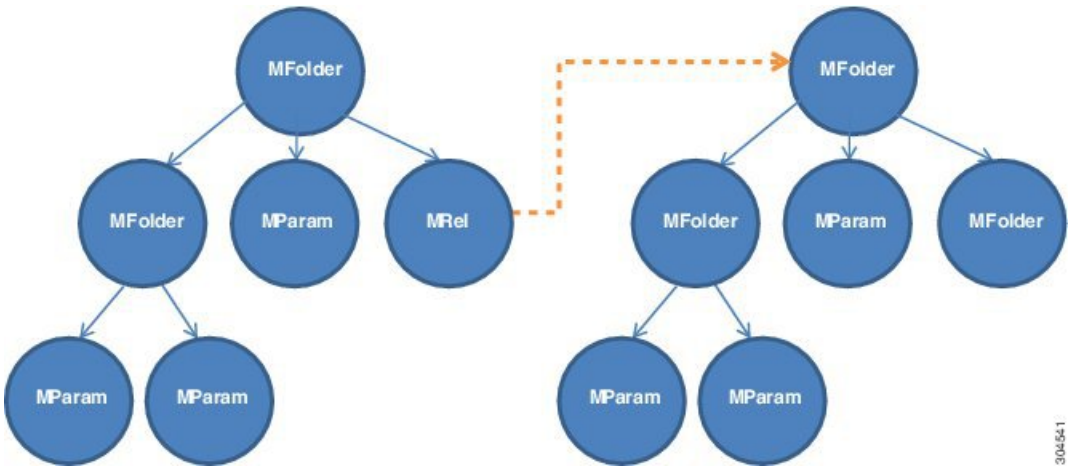
Configuration Parameters Inside the Device Package Specification

A device package contains an XML file that describes the specification for the service device. This specification includes device information as well as various functions provided by the service device.

As part of the device specification, this file must contain the declaration for the configuration needed by the service device. This configuration is needed to configure various functions that are provided by the service device during graph instantiation.

The following figure shows the configuration parameters hierarchy inside the device package.

Figure 1: Configuration Parameters Hierarchy Inside the Device Package



MFolder

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

Attribute	Description
Key	Defines the type of the configuration item. The key is defined in the device package and can never be overwritten. The key is used as a matching criterion as well as for validation.
Description	Describes the configuration item.
Cardinality	Specifies the cardinality of the configuration item. The default value of cardinality is `1`. If cardinality is `N`, The Application Policy Infrastructure Controller (APIC) allows N instances of the configuration parameter to be configured.
ScopedBy	Specifies the scope for the parameter resolution. ScopedBy determines where to look for parameter values when APIC resolves the parameter from configuration MOs. For more information, see #unique_92. Default value is `Epg`. Supported values are `Tenant`, `Ap`, `Bd`, and `Epg`.
RsConnector	A relation that associates a configuration item to an MConn.
DevCtx	Allows a configuration item to be associated with a specific physical device (CDev) in a device cluster (LDev).
Locked	Allows a configuration item value to be locked. Once locked, the value cannot be changed.

MParam

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

Attribute	Description
Key	Defines the type of the configuration item. The key is defined in the device package and can never be overwritten. The key is used as a matching criterion as well as for validation.
Description	Describes the configuration item.
Cardinality	Specifies the cardinality of the configuration item. The default value of cardinality is `1`. If cardinality is `N`, The APIC allows N instances of the configuration parameter to be configured.
RsConnector	A relation that associates a configuration item to an MConn.
Mandatory	Allows a configuration item to be marked as mandatory.
Locked	Allows a configuration item value to be locked. Once locked, the value cannot be changed.
Validation	Specifies the validation method for the value.

MRel

MRel allows one MFolder to refer to another MFolder. Using MRel inside an MFolder, an administrator can associate the containing MFolder to the MFolder that is pointed by the MRel using the RsTarget relation contained inside MRel. MRel has following attributes:

Attribute	Description
Key	Defines the type of the configuration item. The key is defined in the device package and can never be overwritten. The key is used as a matching criterion as well as for validation.
Description	Describes the configuration item.
Cardinality	Specifies the cardinality of the configuration item. The default value of cardinality is `1`.
RsTarget	A relation that associates a configuration folder to another MFolder. The value of TDn for this relation is the DN of the target folder.
RsConnector	A relation that associates a configuration item to an MConn.
Mandatory	Allows a configuration item to be marked as mandatory.

Configuration Scope of a Device Package Specification

In a device specification file, configuration items are arranged in different sections.

MDevCfg

The MDevCfg section describes the device level configuration, which is shared by all service graphs using the device. The Application Policy Infrastructure Controller (APIC) does a reference counting of the configuration objects created by using the configuration items described in this section. Objects are only deleted from a service device after all the graph instances that use the device are deleted.

MFuncCfg

The MFuncCfg describes the configuration that is local to a service function and is specific to a service function. The APIC does a reference counting of the configuration objects created by the configuration items described in this section. Objects are created and deleted whenever a service function is instantiated and deleted.

MGrpCfg

The MGrpCfg describes the configuration that is shared by all functions of a service graph using the device. The APIC does a reference counting of the configuration objects created by using the configuration items described in this section. Objects are only deleted from a service device after all functions from the service graph are deleted.

Example XML of Configuration Parameters Inside the Device Package

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

```
<vnsMFolder key="VServer" scopedBy="epg">
    <vnsRsConnector tDn="uni/infra/mDev-Acme-ADC-1.0/mFunc-SLB/mConn-external"/>
    <vnsMParam key="vservername" description="Name of VServer" mandatory="true"/>
    <vnsMParam key="vip" description="Virtual IP"/>
```

```
    <vnsMParam key="subnet" description="Subnet IP"/>
    <vnsMParam key="port" description="Port for Virtual server"/>
    <vnsMParam key="persistencetype" description="persistencetype"/>
    <vnsMParam key="servicename" description="Service bound to this vServer"/>
    <vnsMParam key="servicetype" description="Service bound to this vServer"/>
    <vnsMParam key="clttimeout" description="Client timeout"/>
    <vnsMFolder key="VServerGlobalConfig" description="This references the global
configuration">
        <vnsMRel key="ServiceConfig">
            <vnsRsTarget tDn="uni/infra/mDev-Acme-ADC-1.0/mDevCfg/mFolder-Service"/>
        </vnsMRel>
        <vnsMRel key="ServerConfig">
            <vnsRsTarget tDn="uni/infra/mDev-Acme-ADC-1.0/mDevCfg/mFolder-Server"/>
        </vnsMRel>
        <vnsMRel key="VipConfig">
            <vnsRsTarget
tDn="uni/infra/mDev-Acme-ADC-1.0/mDevCfg/mFolder-Network/mFolder-vip"/>
            <vnsRsConnector tDn="uni/infra/mDev-Acme-ADC-1.0/mFunc-SLB/mConn-external"/>
        </vnsMRel>
    </vnsMFolder>
</vnsMFolder>
```

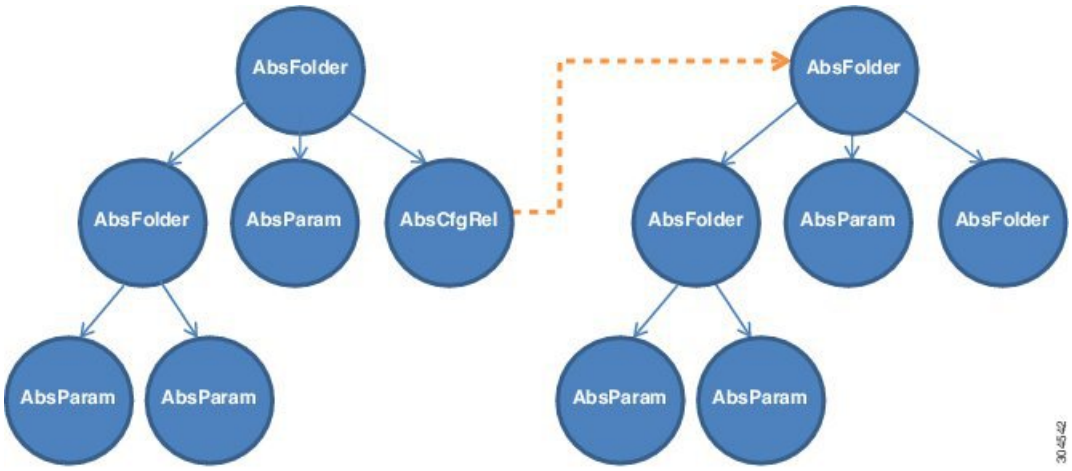
Configuration Parameters Inside An Abstract Function Profile

An abstract profile allows an administrator to configure the default values for the configuration parameters. An abstract function profile contains configuration parameters with values. These values are used as default values during graph instantiation. For more information on how the default values are used, see #unique_92.

An abstract function profile is attached to a function node in a service graph. The default values specified in an abstract function profile is then used when rendering the function onto the service device at the graph instantiation time.

The following figure shows the configuration parameters hierarchy within an abstract function profile.

Figure 2: Configuration Parameters Hierarchy Within an Abstract Function Profile



AbsFolder

AbsFolder is a group of configuration items that can contain AbsParams and other nested AbsFolders. For each AbsFolder, there must be an MFolder inside the device package. The Application Policy Infrastructure Controller (APIC) validates each AbsFolder to ensure that the AbsFolder has a corresponding MFolder in the package. An AbsFolder has the following attributes:

Attribute	Description
Key	Defines the type of the configuration item. The key is defined in the device package and can never be overwritten. The key is used as a matching criterion as well as for validation.
Description	Describes the configuration item.
Cardinality	Specifies the cardinality of the configuration item. The default value is `1`.
ScopedBy	Specifies the scope for the parameter resolution. ScopedBy determines where to look for parameter values when the APIC resolves the parameter from configuration MOs. For more information, see #unique_92. Default value is `Epg`. Supported values are `Tenant`, `Ap`, `Bd`, and `Epg`.
DevCtx	Allows a configuration item to be associated with a specific physical device (CDev) in a device cluster (LDev).
Locked	Allows a configuration item value to be locked. Once locked, the value cannot be changed.

AbsParam

AbsParam is the basic unit of configuration parameters. AbsParam defines a single configuration parameter. As with an AbsFolder, each AbsParam must have an equivalent MFolder in the device specification. The APIC validates the specification to ensure that AbsParam has a corresponding MFolder in the package. The value of an AbsParam is validated using the validation method specified in MParam. An AbsParam has following attributes:

Attribute	Description
Key	Defines the type of the configuration item. The key is defined in the device package and can never be overwritten. The key is used as a matching criterion as well as for validation.
Value	Holds the value for a given configuration item. Values are not supported for MParam.
Description	Describes the configuration item.
Cardinality	Specifies the cardinality of the configuration item. The default value is `1`.
Mandatory	Allows a configuration item to be marked as mandatory.
Locked	Allows a configuration item value to be locked. Once locked, the value cannot be changed.
Validation	Specifies the validation mechanism to be used to validate the configuration parameter.

AbsRel

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

Attribute	Description
Key	Defines the type of the configuration item. The key is defined in the device package and can never be overwritten. The key is used as a matching criterion as well as for validation.
Value	Holds the value for a given configuration item. Values are not supported for MParam.
Description	Describes the configuration item.
Cardinality	Specifies the cardinality of the configuration item. The default value is `1`.
Mandatory	Allows a configuration item to be marked as mandatory.

Configuration Scope of an Abstract Function Profile

Within an abstract function profile, the configuration parameters are structured in a similar manner as to how they are structured inside a device package. There are three different scopes.

AbsDevCfg

This section provides the default value for the configuration items that are declared to be the device level configuration inside of a device package. The configuration items are specified under MDevCfg.

For each configuration item, there must be an equivalent configuration item under the device package.

The configuration that is described in this section is shared by all service graphs that use the device. The Application Policy Infrastructure Controller (APIC) does a reference counting of the configuration objects that are created by using the configuration items that are described in this section. Objects are only deleted from a service device after all graph instances that are using the device are deleted.

AbsGrpCfg

This section provides the default value for the configuration items that are declared to be the device level configuration inside of a device package. The configuration items are specified under MGrpCfg.

For each configuration item, there must be an equivalent configuration item under the device package.

The configuration described in this section is shared by all functions of a service graph that use the device. The APIC does a reference counting of the configuration objects that are created by using the configuration items that are described in this section. Objects are only deleted from a service device after all graph instances that are using the device are deleted.

AbsFuncCfg

This section provides the default value for the configuration items that are declared to be the function level configuration inside of a device package. The configuration items are specified under MFuncCfg.

For each configuration item, there must be an equivalent configuration item under the device package.

This section is used to describe the configuration that is local to a service function. The configuration described in this section is specific to a service function. The APIC does a reference counting of the configuration objects that are created by the configuration items that are described in this section. Objects are created and deleted whenever a service function is instantiated and deleted.

Example XML POST for an Abstract Function Profile With Configuration Parameters

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

```
<vnsAbsFuncProfContr name = "NP">
    <vnsAbsFuncProfGrp name = "Grp1">
        <vnsAbsFuncProf name = "P1">
            <vnsRsProfToMFunc tDn="uni/infra/mDev-Acme-ADC-1.0/mFunc-SLB"/>
            <vnsAbsDevCfg name="D1">
                <vnsAbsFolder key="Service" name="Service-Default" cardinality="n">
                    <vnsAbsParam name="servicetype" key="servicetype" value="TCP"/>
                    <vnsAbsParam name="serviceport" key="serviceport" value="80"/>
                    <vnsAbsParam name="maxclient" key="maxclient" value="1000"/>
                    <vnsAbsParam name="maxreq" key="maxreq" value="100"/>
                    <vnsAbsParam name="cip" key="cip" value="enable"/>
                    <vnsAbsParam name="usip" key="usip" value="enable"/>
                    <vnsAbsParam name="sp" key="sp" value=""/>
                    <vnsAbsParam name="svrtimeout" key="svrtimeout" value="60"/>
                    <vnsAbsParam name="clttimeout" key="clttimeout" value="60"/>
                    <vnsAbsParam name="cka" key="cka" value="NO"/>
                    <vnsAbsParam name="tcpb" key="tcpb" value="NO"/>
                    <vnsAbsParam name="cmp" key="cmp" value="NO"/>
                </vnsAbsFolder>
            </vnsAbsDevCfg>
            <vnsAbsFuncCfg name="SLB">
                <vnsAbsFolder key="VServer" name="VServer-Default">
                    <vnsAbsParam name="port" key="port" value="80"/>
                    <vnsAbsParam name="persistencetype" key="persistencetype"
                      value="cookie"/>
                    <vnsAbsParam name="clttimeout" key="clttimeout" value="100"/>
                    <vnsAbsParam name="servicetype" key="servicetype" value="TCP"/>
                    <vnsAbsParam name="servicename" key="servicename"/>
                </vnsAbsFolder>
            </vnsAbsFuncCfg>
        </vnsAbsFuncProf>
    </vnsAbsFuncProfGrp>
</vnsAbsFuncProfContr>
```

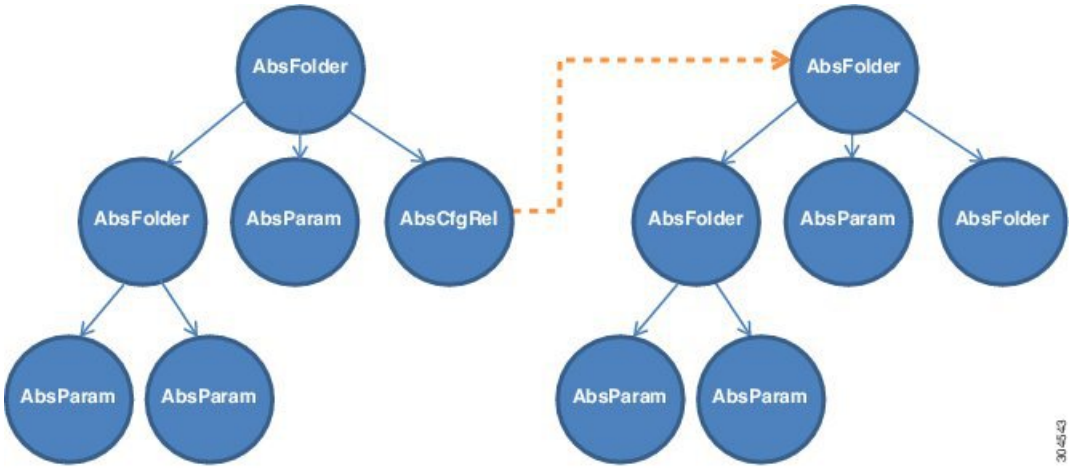
Configuration Parameters Inside an Abstract Function Node in a Service Graph

A function node inside a service graph allows an administrator to configure values for the configuration parameters. These values are used during graph instantiation. For more information on how the values are used, see #unique_92.

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 3: Configuration Parameters Hierarchy Inside an Abstract Function Node



AbsDevCfg

This section is used to provide the default value for the configuration items that are declared to be the device level configuration inside of the device package. The configuration items are specified under MDevCfg.

For each of these configuration items, there must be an equivalent configuration item under the device package.

AbsGrpCfg

This section is used to provide the default value for the configuration items that are declared to be the device level configuration inside of the device package. The configuration items are specified under MGrpCfg.

For each of these configuration items, there must be an equivalent configuration item under the device package.

The configuration that is described in this section is shared by all functions of a service graph that use the device. The Application Policy Infrastructure Controller (APIC) does a reference counting of the configuration objects that are created by using the configuration items that are described in this section. Objects are only deleted from a service device after all functions from the service graph are deleted.

AbsFuncCfg

This section is used to provide the default value for the configuration items that are declared to be the function level configuration inside of the device package. The configuration items are specified under MFuncCfg.

For each of these configuration items, there must be an equivalent configuration item under the device package.

This section is used to describe the configuration that is local to a service function. The configuration that is described in this section is specific to a service function. The APIC does a reference counting of the configuration objects that are created by the configuration items that are described in this section. Objects are created and deleted whenever a service function is instantiated and deleted.

AbsFolder

AbsFolder is a group of configuration items that can contain AbsParamss and other nested AbsFolders. For each AbsFolder, there must be an MFolder inside the device package. The APIC validates each AbsFolder to ensure that the AbsFolder has a corresponding MFolder in the package. AbsFolder has the following attributes:

Attribute	Description
Key	Defines the type of the configuration item. The key is defined in the device package and can never be overwritten. The key is used as a matching criterion as well as for validation.
Description	Describes the configuration item.
Cardinality	Specifies the cardinality of the configuration item. The default value is `1`.
ScopedBy	Specifies the scope for the parameter resolution. ScopedBy determines where to look for parameter values when the APIC resolves the parameter from configuration MOs. For more information, see #unique_92. Default value is `Epg`. Supported values are `Tenant`, `Ap`, `Bd`, and `Epg`.
RsCfgToConn	A relation that associates a configuration item to an AbsConn.
DevCtx	Allows a configuration item to be associated with a specific physical device (CDev) in a device cluster (LDev).
Locked	Allows a configuration item value to be locked. Once locked, the value cannot be changed.

AbsParam

AbsParam is the basic unit of configuration parameters. AbsParam defines a single configuration parameter. As with an AbsFolder, each AbsParam must have an equivalent MFolder in the device specification. The APIC validates the specification to ensure that AbsParam has a corresponding MFolder in the package. The value of an AbsParam is validated using the validation method specified in MParam. AbsParam has following attributes:

Attribute	Description
Key	Defines the type of the configuration item. The key is defined in the device package and can never be overwritten. The key is used as a matching criterion as well as for validation.
Value	Holds the value for a given configuration item. Values are not supported for MParam.
Description	Describes the configuration item.
Cardinality	Specifies the cardinality of the configuration item. The default value is `1`.
RsCfgToConn	A relation that associates a configuration item to an MConn.
Mandatory	Allows a configuration item to be marked as mandatory.
Locked	Allows a configuration item value to be locked. Once locked, the value cannot be changed.
Validation	Specifies the validation mechanism to be used to validate the configuration parameter.

AbsRel

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

Attribute	Description
Key	Defines the type of the configuration item. The key is defined in the device package and can never be overwritten. The key is used as a matching criterion as well as for validation.
Value	Holds the value for a given configuration item. Values are not supported for MParam.
Description	Describes the configuration item.
Cardinality	Specifies the cardinality of the configuration item. The default value is `1`.
RsCfgToConn	A relation that associates a configuration item to an MConn.
Mandatory	Allows a configuration item to be marked as mandatory.
Locked	Allows a configuration item value to be locked. Once locked, the value cannot be changed.

Example XML POST for an Abstract Function Node With Configuration Parameters

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

```
<vnsAbsNode name = "SLB" funcType="GoTo" >
    <vnsRsDefaultScopeToTerm tDn="uni/tn-coke/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-coke/AbsGraph-G3/AbsTermNode-Output1/outtmnl"/>

            <!-- VIP address -->
            <vnsAbsFolder key="vip" name="vip" scopedBy="epg">
                <vnsAbsParam name="vipaddress" key="vipaddress" value=""/>
            </vnsAbsFolder>

            <!-- SNIP address -->
            <vnsAbsFolder key="snip" name="snip" scopedBy="epg">
                <vnsAbsParam name="snipaddress" key="snipaddress" value=""/>
            </vnsAbsFolder>

        </vnsAbsFolder>

        <vnsAbsFolder key="Service" name="Service" scopedBy="epg" cardinality="n">
            <vnsRsScopeToTerm tDn="uni/tn-coke/AbsGraph-G3/AbsTermNode-Output1/outtmnl"/>
            <vnsAbsParam name="servicename" key="servicename" value=""/>
            <vnsAbsParam name="servername" key="servername" value=""/>
            <vnsAbsParam name="serveripaddress" key="serveripaddress" value=""/>
        </vnsAbsFolder>
    </vnsAbsDevCfg>

    <vnsAbsFuncCfg>
        <vnsAbsFolder key="VServer" name="VServer" scopedBy="epg">
            <vnsRsScopeToTerm tDn="uni/tn-coke/AbsGraph-G3/AbsTermNode-Output1/outtmnl"/>
            <!-- Virtual Server Configuration -->
            <vnsAbsParam name="vip" key="vip" value=""/>
            <vnsAbsParam name="vservername" key="vservername" value=""/>
```

```
            <vnsAbsParam name="servicename" key="servicename"/>
            <vnsRsCfgToConn tDn="uni/tn-coke/AbsGraph-G3/AbsNode-Node2/AbsFConn-C4" />
        </vnsAbsFolder>
    </vnsAbsFuncCfg>
    <vnsRsNodeToMFunc tDn="uni/infra/mDev-Acme-ADC-1.0/mFunc-SLB"/>
</vnsAbsNode>
```

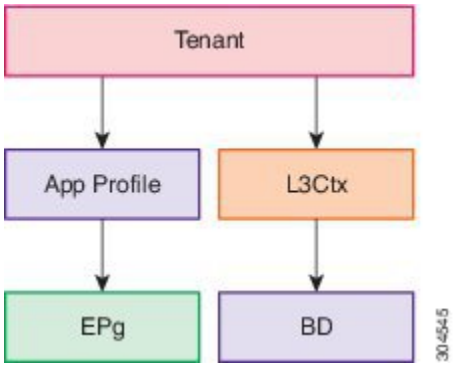
Configuration Parameters Inside Various Configuration MOs

An administrator can specify configuration parameters for a service function as part of various Application Policy Infrastructure Controller (APIC) MOs, such as EPG, tenant, BD, or AP. When a graph is instantiated, the APIC resolves the needed configuration for a graph by looking up the parameters from various places. At instantiation, parameter values are resolved and passed to the device script.

The flexibility of being able to keep configuration parameters inside various MOs allows an administrator to configure a single service graph and then use the graph for different tenants or end point groups (EPGs) with a different configuration for different tenants or EPGs.

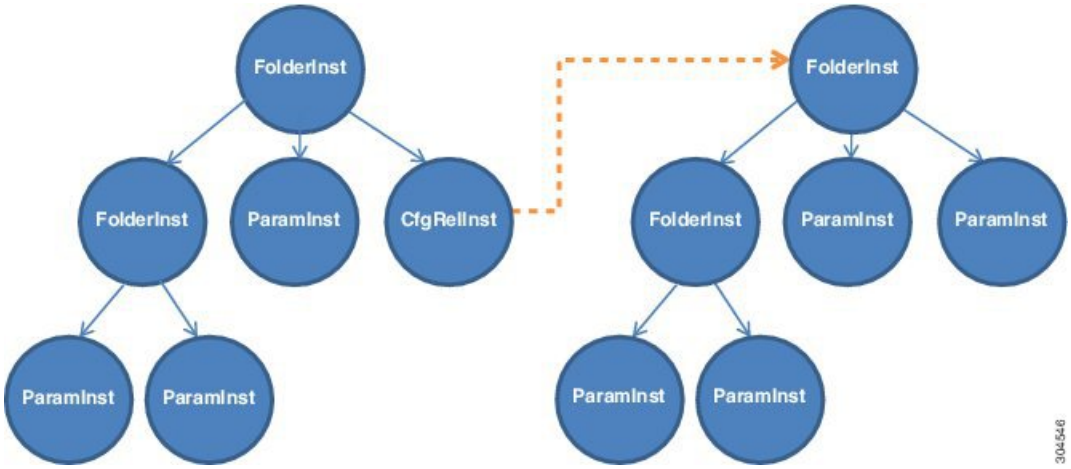
The following figure shows the hierarchy of an APIC MO.

Figure 4: Hierarchy of an APIC MO



The following figure shows configuration parameters inside various configuration MOs.

Figure 5: Configuration Parameters Inside Various Configuration MOs



FolderInst

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

Attribute	Description
Key	Defines the type of the configuration item. The key is defined in the device package and can never be overwritten. The key is used as a matching criterion as well as for validation.
ctrctNameOrLbl	Finds a matching FolderInst during parameter resolution. For a FolderInst to be used for parameter resolution, this property must match with the name of the contract that is associated with the service graph. Otherwise, this FolderInst is skipped and values are not used from this FolderInst. The value of this field can be "any" to allow this FolderInst to be used for all contracts.
graphNameOrLbl	Finds a matching FolderInst during parameter resolution. For a FolderInst to be used for parameter resolution, this property must match with the service graph name. Otherwise, this FolderInst is skipped and values are not used from this FolderInst. The value of this field can be “any” to allow this FolderInst to be used for all service graphs.
nodeNameOrLbl	Finds a matching FolderInst during parameter resolution. For a FolderInst to be used for parameter resolution, this property must match with the node name. Otherwise, this FolderInst is skipped and values are not used from this FolderInst. The value of this field can be “any” to allow this FolderInst to be used for all nodes in a service graph.

ParamInst

ParamInst is the basic unit of configuration parameters. ParamInst defines a single configuration parameter. As with a FolderInst, each ParamInst must have an equivalent MParam in the device specification. The APIC validates the specification to ensure that ParamInst has a corresponding MParam in the package. The value of an ParamInst is validated using the validation method specified in the corresponding MParam. ParamInst has following attributes:

Attribute	Description
Key	Defines the type of the configuration item. The key is defined in the device package and can never be overwritten. The key is used as a matching criterion as well as for validation.
Value	Holds the value for a given configuration item. Values are not supported for MParam.

CfgRelInst

CfgRelInst has following attributes:

Attribute	Description
Key	Defines the type of the configuration item. The key is defined in the device package and can never be overwritten. The key is used as a matching criterion as well as for validation.
Value	Holds the path for the target FolderInst.

Example XML POST for an Application EPG With Configuration Parameters

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

```
<fvAEPg dn="uni/tn-acme/ap-myApp/epg-app"  name="app">

   <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any" nodeNameOrLbl="any" key="Monitor"
 name="monitor1">
      <vnsRsFolderInstToMFolder tDn="uni/infra/mDev-Acme-ADC-1.0/mDevCfg/mFolder-Monitor"/>

       <vnsParamInst name="weight" key="weight" value="10"/>
   </vnsFolderInst>

   <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any" nodeNameOrLbl="any" key="Service"
 name="Service1">
       <vnsParamInst name="servicename" key="servicename" value="crpvgrtst02-8010"/>
       <vnsParamInst name="servicetype" key="servicetype" value="TCP"/>
       <vnsParamInst name="servername" key="servername" value="s192.168.100.100"/>
      <vnsParamInst name="serveripaddress" key="serveripaddress" value="192.168.100.100"/>

       <vnsParamInst name="serviceport" key="serviceport" value="8080"/>
       <vnsParamInst name="svrtimeout" key="svrtimeout" value="9000" />
       <vnsParamInst name="clttimeout" key="clttimeout" value="9000" />
       <vnsParamInst name="usip" key="usip" value="NO" />
       <vnsParamInst name="useproxyport" key="useproxyport" value="" />
       <vnsParamInst name="cip" key="cip" value="ENABLED" />
       <vnsParamInst name="cka" key="cka" value="NO" />
       <vnsParamInst name="sp" key="sp" value="OFF" />
       <vnsParamInst name="cmp" key="cmp" value="NO" />
       <vnsParamInst name="maxclient" key="maxclient" value="0" />
       <vnsParamInst name="maxreq" key="maxreq" value="0" />
       <vnsParamInst name="tcpb" key="tcpb" value="NO" />
       <vnsCfgRelInst name="MonitorConfig" key="MonitorConfig" targetName="monitor1"/>
   </vnsFolderInst>

   <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="G2" nodeNameOrLbl="any" key="Network"
 name="Network">
      <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="G2" nodeNameOrLbl="any" key="vip"
 name="vip">
           <vnsParamInst name="vipaddress1" key="vipaddress" value="10.10.10.200"/>
       </vnsFolderInst>
       <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="G2" nodeNameOrLbl="any"
devCtxLbl="C1" key="snip" name="snip1">
           <vnsParamInst name="snipaddress" key="snipaddress" value="192.168.1.200"/>
       </vnsFolderInst>
       <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="G2" nodeNameOrLbl="any"
devCtxLbl="C2" key="snip" name="snip2">
   <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="G1" nodeNameOrLbl="any" key="Network"
 name="Network">
      <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="G1" nodeNameOrLbl="any" key="vip"
 name="vip">
           <vnsParamInst name="vipaddress1" key="vipaddress" value="10.10.10.100"/>
       </vnsFolderInst>
       <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="G1" nodeNameOrLbl="any"
devCtxLbl="C1" key="snip" name="snip1">
           <vnsParamInst name="snipaddress" key="snipaddress" value="192.168.1.100"/>
       </vnsFolderInst>
       <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="G1" nodeNameOrLbl="any"
devCtxLbl="C2" key="snip" name="snip2">
           <vnsParamInst name="snipaddress" key="snipaddress" value="192.168.1.101"/>
```

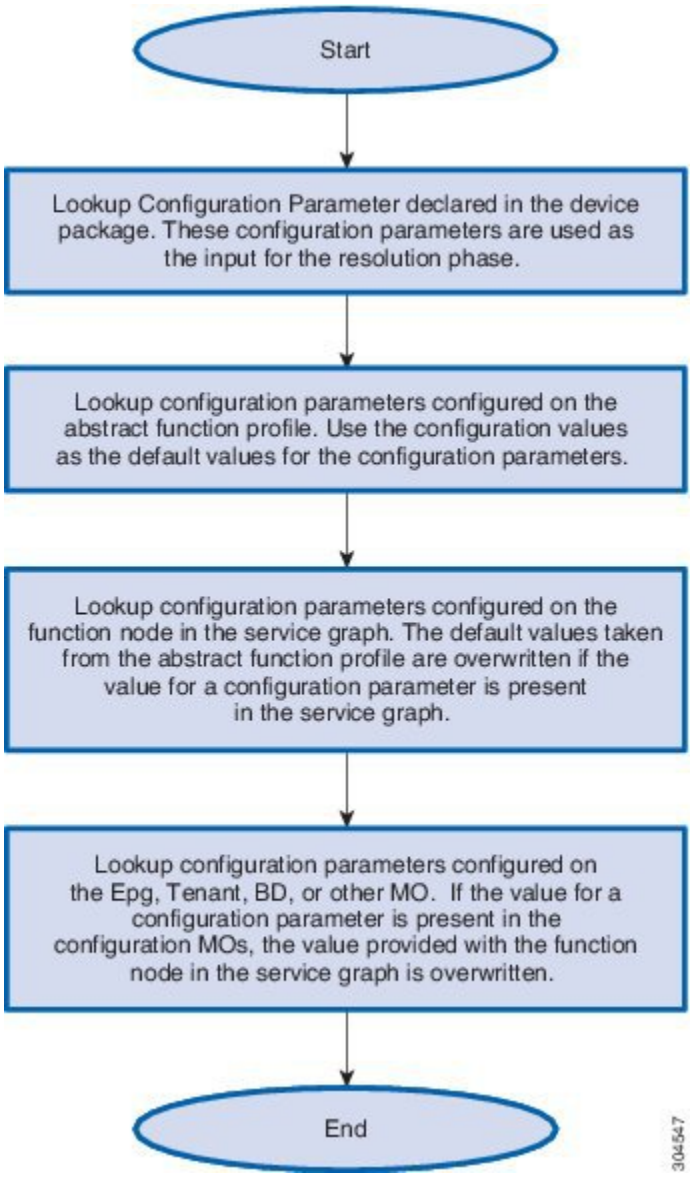
```
        </vnsFolderInst>
        <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="G1" nodeNameOrLbl="any"
devCtxLbl="C3" key="snip" name="snip3">
            <vnsParamInst name="snipaddress" key="snipaddress" value="192.168.1.102"/>
        </vnsFolderInst>
    </vnsFolderInst>

    <!-- SLB Configuration -->
   <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any" nodeNameOrLbl="any" key="VServer"
 name="VServer">
        <!-- Virtual Server Configuration -->
        <vnsParamInst name="port" key="port" value="8010"/>
        <vnsParamInst name="vip" key="vip" value="10.10.10.100"/>
        <vnsParamInst name="vservername" key="vservername" value="crpvgrtst02-vip-8010"/>
        <vnsParamInst name="servicename" key="servicename" value="crpvgrtst02-8010"/>
        <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any" nodeNameOrLbl="any"
key="VServerGlobalConfig" name="VServerGlobalConfig">
           <vnsCfgRelInst name="ServiceConfig" key="ServiceConfig" targetName="Service1"/>

            <vnsCfgRelInst name="VipConfig" key="VipConfig" targetName="Network/vip"/>
        </vnsFolderInst>
    </vnsFolderInst>
</fvAEPg>
```

Parameter Resolution

During graph instantiation, the Application Policy Infrastructure Controller (APIC) resolves the configuration parameters for each function in the service graph. After resolution completes, the parameter values are passed to the device script. The device script uses these parameter values to configure the service on the service appliance.

The following flow chart describes the parameter resolution steps.

Figure 6: Parameter Resolution



Looking Up an MO During Parameter Resolution

The Application Policy Infrastructure Controller (APIC) uses two main constructs while finding the suitable configuration MO to take the configuration parameters from.

RsScopeToTerm

The RsScopeToTerm relation for a function node or for an AbsFolder indicates the terminal node of the service graph that is connected with the configuration MOs that has parameters for the graph. The APIC uses the

configuration MOs that are connected to the specified terminal node in RsScopeToTerm to find the graph configuration parameters.

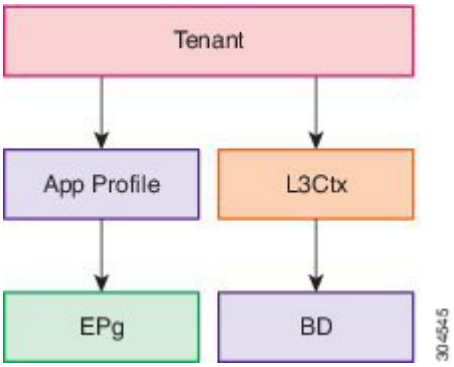
If there is no RsScopeToTerm configuration specified, APIC uses the terminal connected to the provider EPG by default.

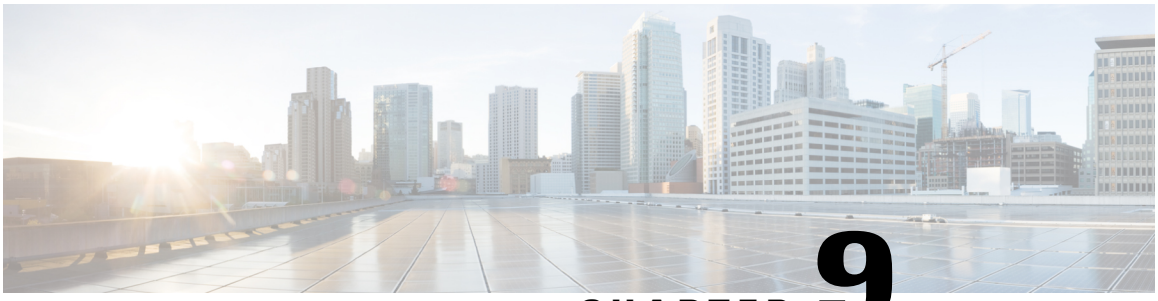
ScopedBy Attribute

The ScopedBy attribute is used to find the starting MO from which to resolve the parameter. For example, if scopedBy has a value of "Epg", the APIC starts the parameter resolution from the endpoint group. The APIC then walks up in the hierarchy to resolve the parameters, walking to the application profile and then to the tenant to resolve the configuration parameters.

The following figure shows the hierarchy of an APIC MO.

Figure 7: Hierarchy of an APIC MO





CHAPTER 9

Using a Service Graph

Using a Service Graph

Associating a Service Graph With a Contract

You can associate a service graph with a contract.

Before You Begin

- Configure a contract. See #unique_103.

Step 1 On the menu bar, click the **TENANTS** tab. The **Tenant** window appears.

Step 2 In the **Navigation** pane, expand the tenant's branch.

Step 3 Expand the **Application Profiles** branch to show the application profiles under that tenant.

Step 4 Expand the branch of an application profile for which you want to associate a service graph with a contract. The list of the application EPGs under that application profile appears.

Step 5 Right click the application EPG of your choice. A pop-up menu appears.

Step 6 Choose **Add Provided Contract**. The **ADD PROVIDED CONTRACT** dialog box appears.

Step 7 From the **Contract** drop-down menu, choose a contract.

Step 8 Click **Submit**. The Application Policy Infrastructure Controller (APIC) connects the application EPG to the service graph's output terminal. The abstract folder/parameters are added from the left pane (L4-L7 Services> Service Graphs> GraphName).

Step 9 You can expand the tree to view and edit the configuration parameters of the function nodes. You can specify the parameter names and parameter values by clicking in the **ABSTRACT FOLDER/PARAM NAME** column and **VALUE**

column and entering the desired information. The abstract folder/parameters are added from the left pane (L4-L7 Services> Service Graphs> GraphName).

Step 10 Click **Submit**. The **ADD PROVIDED CONTRACT** dialog box closes, and the contract is added to the application EPG. You can view the provided contracts in the **Provided Contracts** tab in the **Contracts** dialog box.
Next, you will make that contract a consumer contract on this EPG, which means the contract will be bound between the provider EPG and consumer EPG.

Step 11 Right-click a different EPG. A pop-up menu appears.

Step 12 Choose **Add Consumed Contract**. The **ADD CONSUMED CONTRACT** dialog box appears. This process instantiates the service graph.

Step 13 In the **Contract** drop-down list, choose the contract. The **ADD CONSUMED CONTRACT** dialog box changes to display a **Parameters Configurations** section.

Step 14 You can expand the tree to view and edit the configuration parameters of the consumed contract. You can specify the parameter names and parameter values by clicking in the **ABSTRACT FOLDER/PARAM NAME** column and **VALUE** column and entering the desired information.

Step 15 Click **SUBMIT**. The **ADD CONSUMED CONTRACT** dialog box closes, and the consumed contract is created.

Step 16 In the **Navigation** pane of the **TENANTS** window, expand the tenant's branch.

Step 17 Expand the **Security Policies** branch.

Step 18 Expand the **Contracts** branch to show the contract that you created.

Step 19 Expand the **contract** and click the object under it. In the work pane associate the service graph that was created earlier and click **SUBMIT**.

Step 20 Click **SUBMIT** to attach the contract to the graph.

Configuring Using CLI Commands

An example of a configuration:

```
admin@apic1:contracts> moconfig running


# contract
cd '/aci/tenants/acme/security-policies/contracts'
mocreate 'webCtrct'
cd 'webCtrct'
moset scope 'tenant'
moconfig commit


# contract-subject
cd '/aci/tenants/acme/security-policies/contracts/webCtrct/subjects'
mocreate 'http'
moconfig commit


# subj-graphatt
cd '/aci/tenants/acme/security-policies/contracts/webCtrct/subjects/http'
mocreate subj-graphatt
cd 'subj-graphatt'
moset name 'WebGraph'
moconfig commit


# vz-rssubjfiltatt
cd '/aci/tenants/acme/security-policies/contracts/webCtrct/subjects/http/common-filters'
mocreate 'wildcard'
```

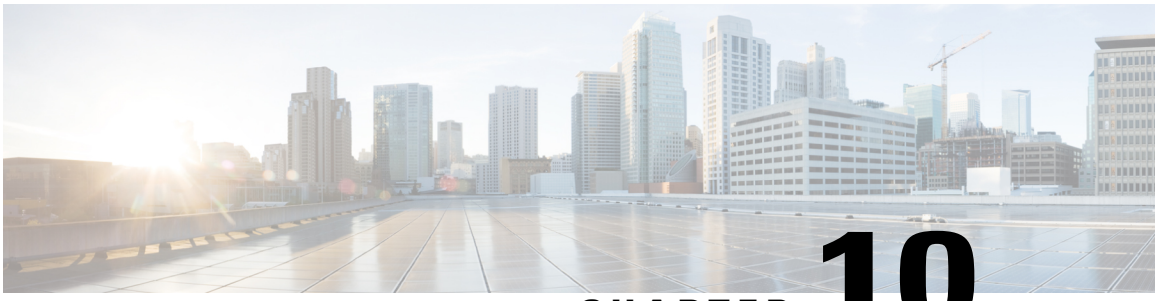
```
moconfig commit
```

Configuring Using REST APIs

```
<polUni>
    <fvTenant dn="uni/tn-acme" name="acme">
<!—L3 Network-->
<fvCtx name="MyNetwork"/>
        <!-- Bridge Domain for MySrvr EPG -->
        <fvBD name="MySrvrBD">
           <fvRsCtx tnFvCtxName="MyNetwork" />
           <fvSubnet ip="10.10.10.10/24">
           </fvSubnet>
</fvBD>
        <!-- Bridge Domain for MyClnt EPG -->
        <fvBD name="MyClntBD">
          <fvRsCtx tnFvCtxName="MyNetwork" />
          <fvSubnet ip="20.20.20.20/24">
          </fvSubnet>
        </fvBD>
        <fvAp dn="uni/tn-acme/ap-MyAP" name="MyAP">
            <fvAEPg dn="uni/tn-acme/ap-MyAP/epg-MyClnt" name="MyClnt">
                <fvRsBd tnFvBDName="MySrvrBD" />
                <fvRsDomAtt tDn="uni/vmmp-Vendor1/dom-MyVMs" />
                <fvRsProv tnVzBrCPName="webCtrct"> </fvRsProv>
<fvRsPathAtt tDn="topology/pod-1/paths-17/pathep-[eth1/21]" encap="vlan-202"/>
<fvRsPathAtt tDn="topology/pod-1/paths-18/pathep-[eth1/21]" encap="vlan-202"/> </fvAEPg>
            <fvAEPg dn="uni/tn-acme/ap-MyAP/epg-MySRVR" name="MySRVR">
                <fvRsBd tnFvBDName="MyClntBD" />
                <fvRsDomAtt tDn="uni/vmmp-Vendor1/dom-MyVMs" />
                <fvRsCons tnVzBrCPName="webCtrct"> </fvRsCons>
<fvRsPathAtt tDn="topology/pod-1/paths-17/pathep-[eth1/21]" encap="vlan-203"/>
<fvRsPathAtt tDn="topology/pod-1/paths-18/pathep-[eth1/21]" encap="vlan-203"/>
            </fvAEPg>
        </fvAp>
    </fvTenant>
</polUni>
```

Configuring Using REST APIs to Create a Security Policy

```
<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>
```



CHAPTER 10

Monitoring a Service Graph

Monitoring a Service Graph

Monitoring a Service Graph Instance

After you configure a service graph 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, click the **TENANTS** tab. The **Tenant** window appears.

Step 2 On the submenu bar, click the tab of the tenant for which you want to monitor its service graph. The Tenant window for the selected tenant appears in the **Work** pane.

Step 3 In the **Navigation** pane, expand the tenant's branch.

Step 4 In the **Navigation** pane, expand the **L4-L7 Services** branch.

Step 5 In the **Navigation** pane, click **Deployed Service Graphs**. The **Work** pane displays the following information about the active service graph instances:

Name	Description
Contract column	The name of the contract that is shown in the service graph.
State column	The state of the service graph. A state of **applied** means that the graph has been applied, and the graph policy is active within the fabric and the service device.
Service Graph column	The name of the service graph.
Contained By column	The name of the network that contains the service graph.

Name	Description
Function Nodes column	The name of the function node that is connected to the graph.

Step 6 Expand the **Deployed Service Graphs** branch. The active service graph instances are listed under the branch.

Step 7 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 8 Expand the branch for one of the graph instances. The functions of the graph instance are listed under the instance.

Step 9 Click one of the functions to view additional information about that function in the **Work** pane. The default view is the policy of that function. You can click one of the tabs in the **Work** pane to change the view for that function. The **Work** pane displays the following information about the policy:

Name	Description
POLICY tab	The function's properties, resources allocated to the function, and the parameters of the function.
OPERATIONAL tab	The health score of the function. The health score is calculated by the Application Policy Infrastructure Controller (APIC), and is based on the information received from the service device about the function.
FAULTS tab	The issues that are happening on the function node.
HISTORY tab	The history of events that occurred on the function node.

Name	Description
POLICY tab	The function's properties, resources allocated to the function, and the parameters of the function.
FAULTS tab	The issues that are happening on the function node.
HISTORY tab	The history of events that occurred on the function node.

Step 10 In the **Navigation** pane, click **Deployed Device Cluster**. The **Work** pane displays information about the device instances.

Monitoring Service Graph Faults

After you configure a service graph and attach the graph to an endpoint group (EPG) and a contract, you can monitor a service graph's faults.

Step 1 On the menu bar, click the **TENANTS** tab. The **Tenant** window appears.

Step 2 On the submenu bar, click the tab of the tenant for which you want to monitor its service graph. The **Tenant** window for the selected tenant appears in the **Work** pane.

Step 3 In the **Navigation** pane, expand the tenant's branch.

Step 4 Expand the **L4-L7 Services** branch.

Step 5 Expand the **Graphs Instances** branch. The active service graph instances are listed under this branch.

Step 6 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 7 Click one of the functions. By default, the **Work** pane shows the policy of that function.

Step 8 Click the **FAULTS** tab in the **Work** pane. The **Work** pane displays the faults of the function node. For information about the faults and how to resolve the faults, see Resolving Service Graph Faults, on page 67.

Resolving Service Graph Faults

After you have observed one or more service graph faults, resolving the issue depends on the fault. The following tables describe the faults and provide how to resolve faults.

Table 1: Connector Faults

Fault	CLI Label	Description and Resolution
missing-connection	connection associated with a connector not found	The configuration for a graph connector is invalid. The associated connection for the connector could not be found.
missing-nodeinst	NodeInst associated with a connector not found	The configuration for a graph connector is invalid. The associated NodeInst for the connector could not be found.
conn-nonrenderable	Graph connector could not be rendered.	The configuration for a graph connector is invalid. The graph could not be rendered.
invalid-bd	BD associated with a connector is not valid	The configuration for a graph connector is invalid. The associated bridge domain for the connector is not valid.

Fault	CLI Label	Description and Resolution
invalid-ctx	Ctx associated with a connector is not valid.	The configuration for a graph connector is invalid. The associated Ctx for the connector is not valid.
missing-peer-conn	Peer connector associated with a connector not found.	Configuration for a graph connector is invalid. The peer connector for the connection could not be found.

Table 2: AbsGraph and GraphInst Faults

Fault	CLI Label	Description and Resolution
invalid-abstract-graph-config	invalid abstract graph config	The abstract graph configuration is invalid.
missing-mandatory-param	mandatory param not found	A mandatory configuration parameter could not be resolved. Check the package for the mandatory parameter and make sure that AbsGraph has the parameter.
param-cardinality-error	invalid param cardinality	A configuration parameter does not meet cardinality requirements. Check if you specified multiple instances of a parameter without specifying cardinality=n.
epp-download-failure	epp download failure	Graph policies failed to download to the switch.
param-duplicate-name-failure	duplicate param name	Multiple identical copies of a parameter were found with the same name.
id-allocation-failure	id allocation failure	A unique network resource (either VLAN or VXLAN) could not be allocated.
missing-ldev	No cluster found	A cluster could not be found.
context-cardinality-violation-failure	invalid cluster context cardinality	The cluster does not support the required tenancy(multi-tenant or single tenant).
function-type-mismatch-failure	invalid function type	The function type is not supported for the selected device cluster. Check if the AbsNode functype and resolved LDevVip function type match.

Fault	CLI Label	Description and Resolution
invalid-abstract-graph-config-param	invalid abstract graph config param	The abstract graph configuration parameter is invalid.
missing-mparam	No parameter definition found	A required parameter definition could not be found.
missing-abs-graph	no abs graph found	The abstract graph configuration is missing for the graph instance.
invalid-param-config	invalid param config	The parameter configuration is invalid.
invalid-param-scope	invalid parameter scope	The parameter scope is invalid. Check the vnsRsScopeToTerm parameter in the AbsGraph to see if parameter is correct.
invalid-ldev	Invalid cluster	The cluster configuration is invalid. Check the status of the resolved LDevVip and correct the fault.
missing-tenant	no tenant found	The tenant could not be found for the graph.
internal-error	internal error	An internal error occurred during graph processing.
resource-allocation-failure	resource allocation failure	A required resource could not be allocated during graph processing.
missing-abs-function	no abstract function found	The abstract function definition is missing.
param-validation-failed	param validation failure	The configuration parameter value is invalid.
missing-mconn	No connector found	A required connector could not be found.
cdev-missing-mgmt-ip	no mgmt ip found for cdev	The management IP address could not be found for concrete device. Check if vnsCMgmt is present for the resolved vnsCDev.
invalid-graphinst	invalid graphinst config	The graph instance is invalid.
missing-interface	no interface found	An interface could not be found.
missing-bd	no bd found	A bridge domain could not be found.
missing-terminal	Terminal node is missing a terminal	Terminal node is missing a terminal. Check the terminal node settings.

Fault	CLI Label	Description and Resolution
missing-namespace	no vlan/vxlan namespace found	The namespace that is needed to allocate the VLAN or VXLAN is missing. Verify that the resolved vnsLDevVip has the phyDomp parameter or the vmmDomp parameter configured that has a relation to the resolved fvnsVlanInstp.
missing-mfunc	No function found in device package	A required function could not be found in the device package. Verify that all AbsNode function types are present in the package.
missing-lif	no cluster interface found	A required cluster interface could not be found. Verify that the vnsLIf parameter in vnsLDevVip is configured correctly.
invalid-absfunc-profile	Abstract Function Profile config is invalid	The abstract function profile configuration is invalid. This fault may be due to an invalid configuration parameter that is specified in the profile.
missing-cdev	No device found	The concrete device could not be found in the cluster. Verify that a valid vnsCDev is present under the resolved vnsLDevVip.
inappropriate-devfolder	Illegal folder in configuration	No corresponding folder was found in the device package.
invalid-devctx	Device context is not legal for this folder	The device package does not allow specifying a device context for this folder.
insufficient-devctx	Folder must have one value for each associated CDev	The folder is concrete device specific. The folder must have at least one value for each concrete device.
cdev-missing-cif	No interface defined	A concrete device must have at least one interface defined.
cdev-missing-pathinfo	Missing path for interface	For a physical service appliance, we must know to which leaf ports the interface is connected. Verify that the vnsCifPathAtt parameter is present for all vnsCIf under the resolved vnsCDev.

Fault	CLI Label	Description and Resolution
missing-cif	Device interfaces does not match cluster	The device interfaces should match the interfaces configured for their cluster. Verify that the vnsCIf parameter and the vnsLIf parameter are present under the resolved vnsLDevVip.
ldevvip-missing-mgmt-ip	No Mgmt ip found for LDevVip	The management IP address could not be found for LDevVip.
lif-invalid-MIf	LIf has an invalid MifLbl	The MIfLbl contained by LIf is not present in the device package.
lif-invalid-CIf	LIf has an invalid CIf	The CIf contained by LIf is not present. Check the concrete device and CIf settings.
missing-function-node	Abstract graph missing function node	An abstract graph must have at least one function node.
graph-loop-detected	Abstract graph config has a loop	The abstract graph configuration is invalid. The configuration has a loop.
gothrough-routing-enabled-both	Both the legs of go through node has routing enabled	Both the legs of the go through node have routing enabled.
invalid-terminal-nodes	Abstract graph has invalid number of terminal nodes	An abstract graph must have at least two terminal nodes.
missing-ldev-ctx	No device context found for LDev	The device context for the device cluster could not be found. Verify that vnsLDevCtx has values that match the contract, graph and node.
arp-flood-enabled	ARP flood is enabled on the management end point group	ARP flood must be disabled for the management endpoint group.
folderinst-validation-failed	FolderInst has key, that is not found in MFolder	The FolderInst's key and value should honor MFolder specifications.
paraminst-validation-failed	ParamInst has key and/or value, that are not found in MParam	ParamInst's key and value should honor MParam specifications.
invalid-mfolder	FolderInst points to an invalid MFolder	FolderInst must point to a valid MFolder.
invalid-mparam	ParamInst points to an invalid MParam	ParamInst must point to a valid MParam.
devfolder-validation-failed	DevFolder has key, that is not found in MFolder	DevFolders key and value should honor MFolder specifications.

Fault	CLI Label	Description and Resolution
devparam-validation-failed	DevParam has key and/or value, that are not found in MParam	DevParam's key and value should honor MParam specifications
cdev-missing-virtual-info	Virtual Object Info is missing in CDev	Virtual object information must be provided if LDevVip is of type Virtual.

Monitoring a Virtual Device

After you configure a service graph 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 device clusters are in use, which VLANs are configured for a device cluster, the parameters passed to the devices in a device cluster, the statistics of the devices in a device cluster, and the health of the devices in a device cluster.

Step 1 On the menu bar, click the **TENANTS** tab. The **Tenant** window appears.

Step 2 On the submenu bar, click the tab of the tenant for which you want to monitor its service graph. The **Tenant** window for the selected tenant appears in the **Work** pane.

Step 3 In the **Navigation** pane, expand the tenant's branch.

Step 4 Expand the **L4-L7 Services** branch.

Step 5 Expand the **Virtual Devices** branch. The virtual devices are listed under this branch.

Step 6 Click one of the virtual devices. By default, the **Work** pane shows the policy of that virtual device. You can click the tabs in the **Work** pane to change the view. The tabs display the following information about the virtual device:

Tab	Description
POLICY tab	The device cluster that is in use, the VLANs that are configured within the device cluster, and the parameters that have been passed to the devices within the device cluster.
OPERATIONAL tab	The statistics that are being received from the various devices in the device cluster.
HEALTH tab	The health of the devices in the device cluster.

Configuring Using CLI Commands

For a service graph:

```
# l4-l7-service-graph
cd '/aci/tenants/acme/l4-l7-services/service-graphs'
mocreate 'WebGraph'
moconfig commit

# consumer-terminal-node
cd '/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/consumer-terminal-nodes'
mocreate 'Output1'
```

```
moconfig commit

# terminal-connector
cd '/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/consumer-terminal-nodes/Output1'
mocreate terminal-connector
cd 'terminal-connector'
moset name 'C6'
moconfig commit

# provider-terminal-node
cd '/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/provider-terminal-nodes'
mocreate 'Input1'
moconfig commit

# terminal-connector
cd '/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/provider-terminal-nodes/Input1'
mocreate terminal-connector
cd 'terminal-connector'
moset name 'C1'
moconfig commit

# function-node
cd '/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/function-nodes'
mocreate 'FW'
cd 'FW'
moset function-type 'gothrough'
moset function-name 'l4-l7-services/packages/CISCO-ASA-1.0.1.8/functions/Firewall'
moconfig commit

# device-config
cd '/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/function-nodes/FW'
mocreate device-config
moconfig commit

# folder
cd '/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/function-nodes/FW/device-config'
mocreate 'ExIntfCfg'
cd 'ExIntfCfg'
moset cardinality 'n'
moset faultcode '0'
moset key 'InterfaceConfig'
moconfig commit

# param
cd
'/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/function-nodes/FW/device-config/ExIntfCfg'
mocreate param 'mask2'
cd 'param-mask2'
moset value '255.255.255.0'
moset faultcode '0'
moset key 'Mask'
moconfig commit

# param
cd
'/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/function-nodes/FW/device-config/ExIntfCfg'
mocreate param 'address2'
cd 'param-address2'
moset value '30.30.30.201'
moset faultcode '0'
moset key 'Address'
moconfig commit

# folder
cd '/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/function-nodes/FW/device-config'
mocreate 'access-list-foo'
cd 'access-list-foo'
moset faultcode '0'
moset key 'AccessList'
moconfig commit

# folder
cd
```

```
'/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/function-nodes/FW/device-config/access-list-foo'
mocreate folder 'entry1'
cd 'folder-entry1'
moset faultcode '0'
moset key 'AccessControlEntry'
moconfig commit

# param
cd
'/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/function-nodes/FW/device-config/access-list-foo/folder-entry1'
mocreate param 'ip'
cd 'param-ip'
moset value 'ip'
moset faultcode '0'
moset key 'protocol'
moconfig commit

# param
cd
'/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/function-nodes/FW/device-config/access-list-foo/folder-entry1'
mocreate param 'order1'
cd 'param-order1'
moset value '10'
moset faultcode '0'
moset key 'order'
moconfig commit

# param
cd
'/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/function-nodes/FW/device-config/access-list-foo/folder-entry1'
mocreate param 'action-permit'
cd 'param-action-permit'
moset value 'permit'
moset faultcode '0'
moset key 'action'
moconfig commit

# folder
cd '/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/function-nodes/FW/device-config'
mocreate 'InIntfCfg'
cd 'InIntfCfg'
moset cardinality 'n'
moset faultcode '0'
moset key 'InterfaceConfig'
moconfig commit

# param
cd
'/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/function-nodes/FW/device-config/InIntfCfg'
mocreate param 'mask1'
cd 'param-mask1'
moset value '255.255.255.0'
moset faultcode '0'
moset key 'Mask'
moconfig commit

# param
cd
'/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/function-nodes/FW/device-config/InIntfCfg'
mocreate param 'address1'
cd 'param-address1'
moset value '40.40.40.102'
moset faultcode '0'
moset key 'Address'
moconfig commit

# function-config
cd '/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/function-nodes/FW'
mocreate function-config
moconfig commit

# folder
cd
```

```
'/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/function-nodes/FW/function-config'
mocreate 'InIntfCfgRelFolder'
cd 'InIntfCfgRelFolder'
moset faultcode '0'
moset key 'InIntfConfigRelFolder'
moconfig commit

# cfg-rel
cd
'/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/function-nodes/FW/function-config/InIntfCfgRelFolder'
mocreate cfg-rel 'InIntfCfgRel'
cd 'cfg-rel-InIntfCfgRel'
moset faultcode '0'
moset key 'InIntfConfigRel'
moset targetname 'InIntfCfg'
moconfig commit

# folder
cd
'/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/function-nodes/FW/function-config'
mocreate 'ExIntfCfgRelFolder'
cd 'ExIntfCfgRelFolder'
moset faultcode '0'
moset key 'ExIntfConfigRelFolder'
moconfig commit

# cfg-rel
cd
'/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/function-nodes/FW/function-config/ExIntfCfgRelFolder'
mocreate cfg-rel 'ExIntfCfgRel'
cd 'cfg-rel-ExIntfCfgRel'
moset faultcode '0'
moset key 'ExIntfConfigRel'
moset targetname 'ExIntfCfg'
moconfig commit

# folder
cd
'/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/function-nodes/FW/function-config'
mocreate 'access-group-foo'
cd 'access-group-foo'
moset faultcode '0'
moset key 'AccessGroup'
moconfig commit

# param
cd
'/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/function-nodes/FW/function-config/access-group-foo'
mocreate param 'dir'
cd 'param-dir'
moset value 'in'
moset faultcode '0'
moset key 'direction'
moconfig commit

# param
cd
'/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/function-nodes/FW/function-config/access-group-foo'
mocreate param 'con1'
cd 'param-con1'
moset value 'external'
moset faultcode '0'
moset key 'connector'
moconfig commit

# cfg-rel
cd
'/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/function-nodes/FW/function-config/access-group-foo'
mocreate cfg-rel 'access-list-foo-rel'
cd 'cfg-rel-access-list-foo-rel'
moset faultcode '0'
moset key 'access_list_name'
moset targetname 'access-list-foo'
```

```
moconfig commit

# function-connector
cd '/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/function-nodes/FW'
mocreate function-connector 'external'
cd 'function-connector-external'
moset ctx-terms 'uni/tn-acme/AbsGraph-WebGraph/AbsTermNodeProv-Input1/intmnl'
moset meta-connector 'uni/infra/mDev-CISCO-ASA-1.0.1.8/mFunc-Firewall/mConn-external'
moconfig commit

# function-connector
cd '/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/function-nodes/FW'
mocreate function-connector 'internal'
cd 'function-connector-internal'
moset ctx-terms 'uni/tn-acme/AbsGraph-WebGraph/AbsTermNodeCon-Output1/outtmnl'
moset meta-connector 'uni/infra/mDev-CISCO-ASA-1.0.1.8/mFunc-Firewall/mConn-internal'
moconfig commit

# default-scope-to-term
cd '/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/function-nodes/FW'
mocreate default-scope-to-term
cd 'default-scope-to-term'
moset target-dn 'uni/tn-acme/AbsGraph-WebGraph/AbsTermNodeCon-Output1/outtmnl'
moconfig commit

# function-node
cd '/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/function-nodes'
mocreate 'LoadBalancing'
cd 'LoadBalancing'
moset function-name 'l4-l7-services/packages/Citrix-NetScaler-10.5/functions/LoadBalancing'
moconfig commit

# function-connector
cd '/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/function-nodes/LoadBalancing'
mocreate function-connector 'inside'
cd 'function-connector-inside'
moset meta-connector 'uni/infra/mDev-Citrix-NetScaler-10.5/mFunc-LoadBalancing/mConn-internal'
moconfig commit

# function-connector
cd '/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/function-nodes/LoadBalancing'
mocreate function-connector 'outside'
cd 'function-connector-outside'
moset meta-connector 'uni/infra/mDev-Citrix-NetScaler-10.5/mFunc-LoadBalancing/mConn-external'
moconfig commit

# connection
cd '/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/connections'
mocreate 'CON2'
moconfig commit

# relation-from-abstract-connection-to-abstract-connectors
cd '/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/connections/CON2'
mocreate
'tenants/acme/l4-l7-services/service-graphs/WebGraph/function-nodes/LoadBalancing/function-connector-outside'
moconfig commit

# relation-from-abstract-connection-to-abstract-connectors
cd '/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/connections/CON2'
mocreate
'tenants/acme/l4-l7-services/service-graphs/WebGraph/function-nodes/FW/function-connector-external'
moconfig commit

# connection
cd '/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/connections'
mocreate 'CON3'
cd 'CON3'
moset unicast-routing 'no'
moconfig commit

# relation-from-abstract-connection-to-abstract-connectors
cd '/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/connections/CON3'
mocreate
```

```
'tenants/acme/l4-l7-services/service-graphs/WebGraph/function-nodes/FW/function-connector-internal'
moconfig commit

# relation-from-abstract-connection-to-abstract-connectors
cd '/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/connections/CON3'
mocreate
'tenants/acme/l4-l7-services/service-graphs/WebGraph/consumer-terminal-nodes/Output1/terminal-connector'
moconfig commit

# connection
cd '/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/connections'
mocreate 'CON1'
cd 'CON1'
moset unicast-routing 'no'
moconfig commit

# relation-from-abstract-connection-to-abstract-connectors
cd '/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/connections/CON1'
mocreate
'tenants/acme/l4-l7-services/service-graphs/WebGraph/function-nodes/LoadBalancing/function-connector-inside'
moconfig commit

# relation-from-abstract-connection-to-abstract-connectors
cd '/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/connections/CON1'
mocreate
'tenants/acme/l4-l7-services/service-graphs/WebGraph/provider-terminal-nodes/Input1/terminal-connector'
moconfig commit
admin@apic1:WebGraph>
moset meta-connector 'uni/infra/mDev-Citrix-NetScaler-10.5/mFunc-LoadBalancing/mConn-external'
moconfig commit

# connection
cd '/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/connections'
mocreate 'CON2'
moconfig commit

# relation-from-abstract-connection-to-abstract-connectors
cd '/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/connections/CON2'
mocreate
'tenants/acme/l4-l7-services/service-graphs/WebGraph/function-nodes/LoadBalancing/function-connector-outside'
moconfig commit

# relation-from-abstract-connection-to-abstract-connectors
cd '/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/connections/CON2'
mocreate
'tenants/acme/l4-l7-services/service-graphs/WebGraph/function-nodes/FW/function-connector-external'
moconfig commit

# connection
cd '/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/connections'
mocreate 'CON3'
cd 'CON3'
moset unicast-routing 'no'
moconfig commit

# relation-from-abstract-connection-to-abstract-connectors
cd '/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/connections/CON3'
mocreate
'tenants/acme/l4-l7-services/service-graphs/WebGraph/function-nodes/FW/function-connector-internal'
moconfig commit

# relation-from-abstract-connection-to-abstract-connectors
cd '/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/connections/CON3'
mocreate
'tenants/acme/l4-l7-services/service-graphs/WebGraph/consumer-terminal-nodes/Output1/terminal-connector'
moconfig commit

# connection
cd '/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/connections'
mocreate 'CON1'
cd 'CON1'
moset unicast-routing 'no'
moconfig commit
```

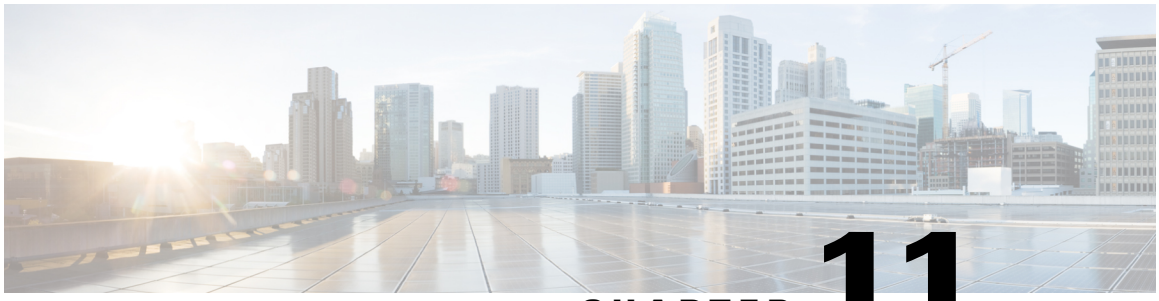
```
# relation-from-abstract-connection-to-abstract-connectors
cd '/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/connections/CON1'
mocreate
'tenants/acme/l4-l7-services/service-graphs/WebGraph/function-nodes/LoadBalancing/function-connector-inside'
moconfig commit

# relation-from-abstract-connection-to-abstract-connectors
cd '/aci/tenants/acme/l4-l7-services/service-graphs/WebGraph/connections/CON1'
mocreate
'tenants/acme/l4-l7-services/service-graphs/WebGraph/provider-terminal-nodes/Input1/terminal-connector'
moconfig commit
```

For a device cluster:

```
# device-cluster
cd '/aci/tenants/acme/l4-l7-services/device-clusters'
mocreate 'Firewall'
cd 'Firewall'
moset function-type 'gothrough'
moset faultcode '0'
moset device-package 'l4-l7-services/packages/CISCO-ASA-1.0.1.8'
moset physical-domain
'fabric/access-policies/physical-and-external-domains/physical-domains/phys'
moset virtual-ip-address '172.23.49.195'
moset port '443'
moset username 'admin'
moconfig commit

# folder
cd '/aci/tenants/acme/l4-l7-services/device-clusters/Firewall/parameters'
mocreate folder 'Timeouts' 'Timeouts'
cd 'folder-Timeouts-Timeouts'
moset faultcode '0'
moconfig commit

# param
cd
'/aci/tenants/acme/l4-l7-services/device-clusters/Firewall/parameters/folder-Timeouts-Timeouts'
mocreate param 'Udp' 'Udp'
cd 'param-Udp-Udp'
moset faultcode '0'
moset value '0:35:0'
moconfig commit

# param
cd
'/aci/tenants/acme/l4-l7-services/device-clusters/Firewall/parameters/folder-Timeouts-Timeouts'
mocreate param 'Connection' 'Connection'
cd 'param-Connection-Connection'
moset faultcode '0'
moset value '0:10:0'
moconfig commit

# logical-interface
cd '/aci/tenants/acme/l4-l7-services/device-clusters/Firewall/logical-interfaces'
mocreate 'external'
cd 'external'
moset type 'uni/infra/mDev-CISCO-ASA-1.0.1.8/mIfLbl-external'
moconfig commit

# vns-rscifatt
cd
'/aci/tenants/acme/l4-l7-services/device-clusters/Firewall/logical-interfaces/external/concrete-interfaces'
mocreate 'tenants/acme/l4-l7-services/device-clusters/Firewall/devices/ASA/interfaces/Gig0_0'
moconfig commit

# logical-interface
cd '/aci/tenants/acme/l4-l7-services/device-clusters/Firewall/logical-interfaces'
mocreate 'internal'
cd 'internal'
moset type 'uni/infra/mDev-CISCO-ASA-1.0.1.8/mIfLbl-internal'
```

```
moconfig commit

# vns-rscifatt
cd
'/aci/tenants/acme/l4-l7-services/device-clusters/Firewall/logical-interfaces/internal/concrete-interfaces'
mocreate 'tenants/acme/l4-l7-services/device-clusters/Firewall/devices/ASA/interfaces/Gig0_1'
moconfig commit

# concrete-device
cd '/aci/tenants/acme/l4-l7-services/device-clusters/Firewall/devices'
mocreate 'ASA'
cd 'ASA'
moset faultcode '0'
moset management-address '172.23.49.195'
moset management-port '443'
moset username 'admin'
moconfig commit

# folder
cd '/aci/tenants/acme/l4-l7-services/device-clusters/Firewall/devices/ASA/parameters'
mocreate folder 'appInsp' 'ApplicationInspection'
cd 'folder-appInsp-ApplicationInspection'
moset faultcode '0'
moconfig commit

# param
cd
'/aci/tenants/acme/l4-l7-services/device-clusters/Firewall/devices/ASA/parameters/folder-appInsp-ApplicationInspection'
mocreate param 'icmp' 'icmp'
cd 'param-icmp-icmp'
moset faultcode '0'
moset value 'enable'
moconfig commit

# concrete-interface
cd '/aci/tenants/acme/l4-l7-services/device-clusters/Firewall/devices/ASA/interfaces'
mocreate 'Gig0_0'
cd 'Gig0_0'
moset path 'topology/pod-1/paths-101/pathep-[eth1/23]'
moconfig commit

# concrete-interface
cd '/aci/tenants/acme/l4-l7-services/device-clusters/Firewall/devices/ASA/interfaces'
mocreate 'Gig0_1'
cd 'Gig0_1'
moset path 'topology/pod-1/paths-101/pathep-[eth1/25]'
moconfig commit

# device-cluster
cd '/aci/tenants/acme/l4-l7-services/device-clusters'
mocreate 'ADCCluster1'
cd 'ADCCluster1'
moset device-type 'virtual'
moset faultcode '0'
moset device-package 'l4-l7-services/packages/Citrix-NetScaler-10.5'
moset vmm-domain 'vm-networking/policies/vmware/vmm-domains/mininet'
moset virtual-ip-address '172.23.49.175'
moset port '80'
moset username 'nsroot'
moconfig commit

# logical-interface
cd '/aci/tenants/acme/l4-l7-services/device-clusters/ADCCluster1/logical-interfaces'
mocreate 'outside'
cd 'outside'
moset type 'uni/infra/mDev-Citrix-NetScaler-10.5/mIfLbl-outside'
moconfig commit

# vns-rscifatt
cd
'/aci/tenants/acme/l4-l7-services/device-clusters/ADCCluster1/logical-interfaces/outside/concrete-interfaces'
mocreate 'tenants/acme/l4-l7-services/device-clusters/ADCCluster1/devices/ADC1/interfaces/1_1'
moconfig commit
```

```
# logical-interface
cd '/aci/tenants/acme/l4-l7-services/device-clusters/ADCCluster1/logical-interfaces'
mocreate 'inside'
cd 'inside'
moset type 'uni/infra/mDev-Citrix-NetScaler-10.5/mIfLbl-inside'
moconfig commit

# vns-rscifatt
cd
'/aci/tenants/acme/l4-l7-services/device-clusters/ADCCluster1/logical-interfaces/inside/concrete-interfaces'
mocreate 'tenants/acme/l4-l7-services/device-clusters/ADCCluster1/devices/ADC1/interfaces/1_2'
moconfig commit

# concrete-device
cd '/aci/tenants/acme/l4-l7-services/device-clusters/ADCCluster1/devices'
mocreate 'ADC1'
cd 'ADC1'
moset context-label 'C1'
moset vm-name 'NSVPX-ESX-OVF'
moset vcenter-name 'vcenter1'
moset faultcode '0'
moset management-address '172.23.49.175'
moset management-port '80'
moset username 'nsroot'
moconfig commit

# concrete-interface
cd '/aci/tenants/acme/l4-l7-services/device-clusters/ADCCluster1/devices/ADC1/interfaces'
mocreate '1_1'
cd '1_1'
moset vnic 'Network adapter 1'
moconfig commit

# concrete-interface
cd '/aci/tenants/acme/l4-l7-services/device-clusters/ADCCluster1/devices/ADC1/interfaces'
mocreate '1_2'
cd '1_2'
moset vnic 'Network adapter 2'
moconfig commit
```



CHAPTER 11

Configuring Administrator Roles for Managing a Service Configuration

Configuring Administrator Roles for Managing a Service Configuration

About Privileges

You can grant privileges to the roles that you configure in the Application Policy Infrastructure Controller (APIC). Privileges determine what tasks a role is allowed to perform. You can grant the following privileges to the administrator roles:

Privilege	Description
nw-svc-policy	The network service policy privilege enables you to do the following: • Create a service graph • Attach a service graph to an application endpoint group (EPG) and a contract • Monitor a service graph
nw-svc-device	The network service device privilege enables you to do the following: • Create a device cluster • Create a concrete device • Create a device context

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 Management

To enable a role to manage service graphs, you must grant the following privilege to that role:

- nw-svc-policy

Configuring a Role for Uploading Device Package

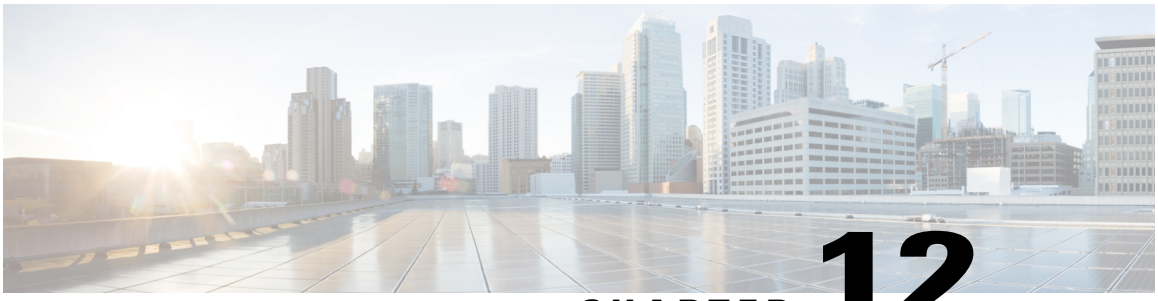
A device package can be uploaded only with the APIC infra admin privilege. Infra admin uploads the device packages. All other tenant administrators have read-only access to the device package. Tenant administrators can access and use various functions available from the device package.

Configuring a Role for Exporting Device Cluster

Device clusters can be exported to enable sharing of device clusters among tenants. A tenant with the role **nw-device** can create a device cluster. If the tenant that owns the device cluster 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 device clusters.

Note To be able to use imported device clusters, other tenants that have imported device clusters need to have the **nw-svc-policy** privilege.



CHAPTER 12

Developing Automation

Developing Automation

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 or Python scripts 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 use the monitoring as 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.

The *Cisco APIC REST API User Guide* contains a list of the northbound REST APIs, along with information on how you can use the APIs. You can write your own orchestration tools using these APIs to configure your services on the APIC.

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

For information on how to construct POSTs using the northbound REST API, including information how to do so in Python, see the *Cisco APIC REST API User Guide*.

Examples of Using the REST APIs

The following REST request creates a tenant with a broadcast domain, an L3 network, application EPGs, and an application profile:

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

        <!—L3 Network-->
        <fvCtx name="MyNetwork"/>

        <!-- Bridge Domain for MySrvr EPG -->
        <fvBD name="MySrvrBD">
           <fvRsCtx tnFvCtxName="MyNetwork" />
           <fvSubnet ip="10.10.10.10/24">
           </fvSubnet>
        </fvBD>

        <!-- Bridge Domain for MyClnt EPG -->
        <fvBD name="MyClntBD">
          <fvRsCtx tnFvCtxName="MyNetwork" />
          <fvSubnet ip="20.20.20.20/24">
          </fvSubnet>
        </fvBD>

        <fvAp dn="uni/tn-acme/ap-MyAP" name="MyAP">

            <fvAEPg dn="uni/tn-acme/ap-MyAP/epg-MyClnt" name="MyClnt">
                <fvRsBd tnFvBDName="MySrvrBD" />
                <fvRsDomAtt tDn="uni/vmmp-Vendor1/dom-MyVMs" />
                <fvRsProv tnVzBrCPName="webCtrct"> </fvRsProv>
              <fvRsPathAtt tDn="topology/pod-1/paths-17/pathep-[eth1/21]" encap="vlan-202"/>

              <fvRsPathAtt tDn="topology/pod-1/paths-18/pathep-[eth1/21]" encap="vlan-202"/>

            </fvAEPg>

            <fvAEPg dn="uni/tn-acme/ap-MyAP/epg-MySRVR" name="MySRVR">
                <fvRsBd tnFvBDName="MyClntBD" />
                <fvRsDomAtt tDn="uni/vmmp-Vendor1/dom-MyVMs" />
                <fvRsCons tnVzBrCPName="webCtrct"> </fvRsCons>
              <fvRsPathAtt tDn="topology/pod-1/paths-17/pathep-[eth1/21]" encap="vlan-203"/>

              <fvRsPathAtt tDn="topology/pod-1/paths-18/pathep-[eth1/21]" encap="vlan-203"/>

            </fvAEPg>
        </fvAp>
    </fvTenant>
</polUni>
```

The following REST request creates a VLAN namespace:

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

The following REST request creates a VMM domain:

```
<polUni>
    <vmmProvP vendor="Vendor1">
        <vmmDomP name="MyVMs">
            <infraRsVlanNs tDn="uni/infra/vlanns-MyNS-dynamic" />
            <vmmUsrAccP name="admin" usr="administrator" pwd="in$1eme" />
            <vmmCtrlrP name="vcenter1" hostOrIp="192.168.64.186">
                <vmmRsAcc tDn="uni/vmmp-Vendor1/dom-MyVMs/usracc-admin" />
            </vmmCtrlrP>
        </vmmDomP>
    </vmmProvP>
</polUni>
```

The following REST request creates a physical domain:

```
<polUni>
    <physDomP name="phys">
        <infraRsVlanNs tDn="uni/infra/vlanns-MyNS-dynamic"/>
    </physDomP>
</polUni>
```

The following REST request creates a device cluster:

```
<polUni>
    <fvTenant dn="uni/tn-acme" name="acme">
        <vnsLDevVip name="ADCCluster1" contextAware=1>
            <vnsRsMDevAtt tDn="uni/infra/mDev-Acme-ADC-1.0"/>
            <vnsRsDevEpg tDn="uni/tn-acme/ap-services/epg-ifc"/>
            <vnsRsALDevToPhysDomP tDn="uni/phys-phys"/>

            <vnsCMgmt name="devMgmt" host="42.42.42.100" port="80" />

            <vnsCCred name="username"value="admin"/>

            <vnsCCredSecret name="password" value="admin" />
        </vnsLDevVip>
    </fvTenant>
</polUni>
```

The following REST request creates a device cluster context:

```
<polUni>
    <fvTenant dn="uni/tn-acme" name="acme">
        <vnsLDevCtx ctrctNameOrLbl="webCtrct" graphNameOrLbl="G1" nodeNameOrLbl="Node1">
            <vnsRsLDevCtxToLDev tDn="uni/tn-acme/lDevVip-ADCCluster1"/>
            <vnsLIfCtx connNameOrLbl="ssl-inside">
                <vnsRsLIfCtxToLIf tDn="uni/tn-acme/lDevVip-ADCCluster1/lIf-int"/>
            </vnsLIfCtx>
            <vnsLIfCtx connNameOrLbl="any">
                <vnsRsLIfCtxToLIf tDn="uni/tn-acme/lDevVip-ADCCluster1/lIf-ext"/>
            </vnsLIfCtx>
        </vnsLDevCtx>
    </fvTenant>
</polUni>
```

The following REST request adds a logical interface in a device cluster:

```
<polUni>
    <fvTenant dn="uni/tn-acme" name="acme">
        <vnsLDevVip name="ADCCluster1">
            <vnsLIf name="C5">
                <vnsRsMetaIf tDn="uni/infra/mDev-Acme-ADC-1.0/mIfLbl-outside"/>
                <vnsRsCIfAtt tDn="uni/tn-acme/lDevVip-ADCCluster1/cDev-ADC1/cIf-int"/>
            </vnsLIf>
            <vnsLIf name="C4">
                <vnsRsMetaIf tDn="uni/infra/mDev-Acme-ADC-1.0/mIfLbl-inside"/>
                <vnsRsCIfAtt tDn="uni/tn-acme/lDevVip-ADCCluster1/cDev-ADC1/cIf-ext"/>
            </vnsLIf>
        </vnsLDevVip>
    </fvTenant>
</polUni>
```

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

```
<polUni>
    <fvTenant dn="uni/tn-acme" name="acme">
        <vnsLDevVip name="ADCCluster1">
            <vnsCDev name="ADC1" devCtxLbl="C1">
                <vnsCIf name="int">
                    <vnsRsCIfPathAtt tDn="topology/pod-1/paths-17/pathep-[eth1/22]"/>
                </vnsCIf>
                <vnsCIf name="ext">
                    <vnsRsCIfPathAtt tDn="topology/pod-1/paths-17/pathep-[eth1/21]"/>
                </vnsCIf>
                <vnsCIf name="mgmt">
                    <vnsRsCIfPathAtt tDn="topology/pod-1/paths-17/pathep-[eth1/20]"/>
                </vnsCIf>
                <vnsCMgmt name="devMgmt" host="172.30.30.100" port="80"/>
                <vnsCCred name="username" value="admin" />
                <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>
                <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>
        </vnsLDevVip>
    </fvTenant>
</polUni>
```

The following REST request creates a service graph:

```
<polUni>
    <fvTenant name="acme">
        <vnsAbsGraph name = "G1">

        <vnsAbsTermNode name = "Input1">
            <vnsAbsTermConn name = "C1" direction = "output">
            </vnsAbsTermConn>
        </vnsAbsTermNode>

        <!-- Node1 Provides SLB functionality -->
        <vnsAbsNode name = "Node1" funcType="GoTo" >
            <vnsRsDefaultScopeToTerm
                tDn="uni/tn-acme/AbsGraph-G1/AbsTermNode-Output1/outtmnl"/>

            <vnsAbsFuncConn name = "C4" direction = "input">
                <vnsRsMConnAtt tDn="uni/infra/mDev-Acme-ADC-1.0/mFunc-SLB/mConn-external"
/>
                <vnsRsConnToLIf tDn="uni/tn-acme/lDevVip-ADCCluster1/lIf-C4" />
            </vnsAbsFuncConn>

            <vnsAbsFuncConn name = "C5" direction = "output">
                <vnsRsMConnAtt tDn="uni/infra/mDev-Acme-ADC-1.0/mFunc-SLB/mConn-internal"
/>
                <vnsRsConnToLIf tDn="uni/tn-acme/lDevVip-ADCCluster1/lIf-C5" />
            </vnsAbsFuncConn>

            <vnsRsNodeToMFunc tDn="uni/infra/mDev-Acme-ADC-1.0/mFunc-SLB"/>
        </vnsAbsNode>

        <vnsAbsTermNode name = "Output1">
            <vnsAbsTermConn name = "C6" direction = "input">
            </vnsAbsTermConn>
        </vnsAbsTermNode>
```

```
        <vnsAbsConnection name = "CON1">
            <vnsRsAbsConnectionConns
                tDn="uni/tn-acme/AbsGraph-G1/AbsTermNode-Input1/AbsTConn" />
            <vnsRsAbsConnectionConns
                tDn="uni/tn-acme/AbsGraph-G1/AbsNode-Node1/AbsFConn-C4" />
        </vnsAbsConnection>

        <vnsAbsConnection name = "CON3">
            <vnsRsAbsConnectionConns
                tDn="uni/tn-acme/AbsGraph-G1/AbsNode-Node1/AbsFConn-C5" />
            <vnsRsAbsConnectionConns
                tDn="uni/tn-acme/AbsGraph-G1/AbsTermNode-Output1/AbsTConn" />
        </vnsAbsConnection>
    </vnsAbsGraph>
  </fvTenant>
</polUni>
```

The following REST request creates a security policy (contract):

```
<polUni>
    <fvTenant dn="uni/tn-acme" name="acme">
        <vzFilter name="HttpIn">
            <vzEntry name="e1" prot="6" dToPort="80"/>
        </vzFilter>

        <vzBrCP name="webCtrct">
            <vzSubj name="http">
                <vzRsSubjFiltAtt tnVzFilterName="HttpIn"/>
            </vzSubj>
        </vzBrCP>
    </fvTenant>
</polUni>
```

The following REST request provides graph configuration parameters from an application EPG:

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

      <!-- Application Profile -->
      <fvAp dn="uni/tn-acme/ap-MyAP" name="MyAP">

          <!-- EPG 1  -->
          <fvAEPg dn="uni/tn-acme/ap-MyAP/epg-MyClnt" name="MyClnt">
              <fvRsBd tnFvBDName="MyClntBD" />
              <fvRsDomAtt tDn="uni/vmmp-Vendor1/dom-MyVMs" />
              <fvRsProv tnVzBrCPName="webCtrct">
              </fvRsProv>
            <fvRsPathAtt tDn="topology/pod-1/paths-17/pathep-[eth1/20]" encap="vlan-201"/>

              <fvSubnet name="SrcSubnet" ip="192.168.10.1/24" />
          </fvAEPg>

          <!-- EPG 2 -->
          <fvAEPg dn="uni/tn-acme/ap-MyAP/epg-MySRVR"  name="MySRVR">
             <fvRsBd tnFvBDName="MyClntBD" />
             <fvRsDomAtt tDn="uni/vmmp-Vendor1/dom-MyVMs" />
             <fvRsCons tnVzBrCPName="webCtrct">
             </fvRsCons>

             <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any" nodeNameOrLbl="any"
                 key="Monitor" name="monitor1">
                 <vnsParamInst name="weight" key="weight" value="10"/>
             </vnsFolderInst>

             <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any" nodeNameOrLbl="any"
                 key="Service" name="Service1">
                 <vnsParamInst name="servicename" key="servicename"
                     value="crpvgrtst02-8010"/>
                 <vnsParamInst name="servicetype" key="servicetype" value="TCP"/>
                 <vnsParamInst name="servername" key="servername"
                     value="s192.168.100.100"/>
                 <vnsParamInst name="serveripaddress" key="serveripaddress"
```

```
                    value="192.168.100.100"/>
                <vnsParamInst name="serviceport" key="serviceport" value="8080"/>
                <vnsParamInst name="svrtimeout" key="svrtimeout" value="9000" />
                <vnsParamInst name="clttimeout" key="clttimeout" value="9000" />
                <vnsParamInst name="usip" key="usip" value="NO" />
                <vnsParamInst name="useproxyport" key="useproxyport" value="" />
                <vnsParamInst name="cip" key="cip" value="ENABLED" />
                <vnsParamInst name="cka" key="cka" value="NO" />
                <vnsParamInst name="sp" key="sp" value="OFF" />
                <vnsParamInst name="cmp" key="cmp" value="NO" />
                <vnsParamInst name="maxclient" key="maxclient" value="0" />
                <vnsParamInst name="maxreq" key="maxreq" value="0" />
                <vnsParamInst name="tcpb" key="tcpb" value="NO" />
                <vnsCfgRelInst name="MonitorConfig" key="MonitorConfig"
                    targetName="monitor1"/>
            </vnsFolderInst>

            <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any" nodeNameOrLbl="any"
                key="Network" name="Network">
             <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any" nodeNameOrLbl="any"

                    key="vip" name="vip">
                 <vnsParamInst name="vipaddress1" key="vipaddress" value="10.10.10.100"/>

                </vnsFolderInst>
             <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any" nodeNameOrLbl="any"

                    devCtxLbl="C1" key="snip" name="snip1">
                    <vnsParamInst name="snipaddress" key="snipaddress"
                        value="192.168.1.100"/>
                </vnsFolderInst>
             <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any" nodeNameOrLbl="any"

                    devCtxLbl="C2" key="snip" name="snip2">
                    <vnsParamInst name="snipaddress" key="snipaddress"
                        value="192.168.1.101"/>
                </vnsFolderInst>
             <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any" nodeNameOrLbl="any"

                    devCtxLbl="C3" key="snip" name="snip3">
                    <vnsParamInst name="snipaddress" key="snipaddress"
                        value="192.168.1.102"/>
                </vnsFolderInst>
            </vnsFolderInst>

            <!-- SLB Configuration -->
            <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any" nodeNameOrLbl="any"
                key="VServer" name="VServer">
                <!-- Virtual Server Configuration -->
                <vnsParamInst name="port" key="port" value="8010"/>
                <vnsParamInst name="vip" key="vip" value="10.10.10.100"/>
                <vnsParamInst name="vservername" key="vservername"
                    value="crpvgrtst02-vip-8010"/>
                <vnsParamInst name="servicename" key="servicename"
                    value="crpvgrtst02-8010"/>
                <vnsParamInst name="servicetype" key="servicetype" value="TCP"/>
             <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any" nodeNameOrLbl="any"

                    key="VServerGlobalConfig" name="VServerGlobalConfig">
                    <vnsCfgRelInst name="ServiceConfig" key="ServiceConfig"
                        targetName="Service1"/>
                    <vnsCfgRelInst name="VipConfig" key="VipConfig"
                        targetName="Network/vip"/>
                </vnsFolderInst>
            </vnsFolderInst>
        </fvAEPg>
    </fvAp>
  </fvTenant>
</polUni>
```

The following REST request attaches a service graph to a contract:

```
<polUni>
```

```
    <fvTenant name="acme">
        <vzBrCP name="webCtrct">
            <vzSubj name="http">
                <vzRsSubjGraphAtt graphName="G1" termNodeName="Input1"/>
            </vzSubj>
        </vzBrCP>
    </fvTenant>
</polUni>
```