



Cisco APIC REST API Configuration Guide, Release 4.1(x)

First Published: 2019-03-18

Last Modified: 2024-01-19

Americas Headquarters

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

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

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

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

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

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

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

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

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

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

© 2019–2024 Cisco Systems, Inc. All rights reserved.



CONTENTS

Preface xxiii

- Audience xxiii
- Document Conventions xxiii
- Related Documentation xxiv
- Documentation Feedback xxv

CHAPTER 1

New and Changed Information 1

- New and Changed Information 1

PART I

Part 1: Cisco APIC REST API Usage Guidelines 3

CHAPTER 2

Using the REST API 5

- About the REST API 5
 - Management Information Model 6
 - Object Naming 7
 - Guidelines and Limitations for Using the REST API 8
- Composing REST API Requests 8
 - Read and Write Operations and Filters 8
 - Using Classes in REST API Commands 11
 - Using Managed Objects in REST API Commands 11
 - Creating the API Command 12
 - Composing the API Command Body 14
 - Composing the API Command Body to Call a Method 15
 - Composing the API Command Body for an API Operation on an MO 16
 - Using Tags and Alias 16
- Composing REST API Queries 18

Composing Query Filter Expressions	18
Applying Query Scoping Filters	20
Filtering API Query Results	23
Filter Conditional Operators	23
Sorting and Paginating Query Results	24
Subscribing to Query Results	25
REST API Examples	27
Information About the API Examples	27
Example: Using the JSON API to Add a Leaf Port Selector Profile	27
Example: Using the JSON API to Get Information About a Node	30
Example: Using the JSON API to Get Running Firmware	31
Example: Using the JSON API to Get Top Level System Elements	32
Example: Using the XML API and OwnerTag to Add Audit Log Information to Actions	33
Example: XML Get Endpoints (Devices) with IP and MAC Addresses	34
Example: Monitoring Using the REST API	34
Accessing the REST API	35
Accessing the REST API	35
Invoking the API	35
Configuring the HTTP Request Method and Content Type	35
Configuring HTTP and HTTPS Using the GUI	36
Configuring HTTP and HTTPS Throttling Using the CLI	36
Configuring a Custom Certificate for Cisco ACI HTTPS Access Using the GUI	37
Authenticating and Maintaining an API Session	39
Requiring a Challenge Token for an API Session	41
Logging In	42
Changing Your Own User Credentials	42
REST API Tools	45
Management Information Model Reference	45
Viewing an API Interchange in the GUI	47
Testing the API Using Browser Add-Ons	48
Testing the API with cURL	49
Cisco APIC Python SDK	49
Using the Managed Object Browser (Visore)	50
Visore Browser Page	50

Accessing Visore	51
Running a Query in Visore	52

PART II**Part 2: Common APIC Tasks Using the REST API 53****CHAPTER 3****Managing APIC Using the REST API 55**

Adding Management Access	55
In-Band and Out-of-Band Management Access	55
About Static Management Access	55
Configuring In-Band Management Access Using the REST API	56
Configuring Static In-Band Management Access Using the REST API	59
Configuring Out-of-Band Management Access Using the REST API	61
Configuring Static Out-of-Band Management Access Using the REST API	63
Managing Configuration Files	65
Overview	65
Backing Up, Restoring, and Rolling Back Configuration Files Workflow	65
About Configuration Export to Controllers	66
About Configuration Import to Controller	67
Configuration File Encryption	68
About the fileRemotePath Object	69
Configuring a Remote Location Using the REST API	69
Configuring Configuration File Export to Controller Using the REST API	70
Configuring a Configuration File Import Policy Using the REST API	70
Encrypting Configuration Files Using the REST API	71
Snapshots and Rollbacks	71
Snapshots	71
About Rollbacks	71
Uploading and Downloading Snapshots Using the REST API	72
Configuring and Executing a Configuration Rollback Using the REST API	73
Using Configuration Zones	73
Configuration Zones	73
Configuration Zone Supported Policies	74
Creating Configuration Zones Using the REST API	77

CHAPTER 4	Managing Roles, Users, and Signature-Based Transactions	79
	Managing APIC Roles and Users	79
	User Access, Authorization, and Accounting	79
	Accounting	79
	Multiple Tenant Support	80
	User Access: Roles, Privileges, and Security Domains	80
	Configuring a Custom Role Using the REST API	82
	Configuring a Local User	82
	Configuring a Local User Using the REST API	82
	Configuring a Remote User	83
	Configuring a Remote User Using the REST API	83
	APIC Signature-Based Transactions	84
	About Signature-Based Transactions	84
	Using a Private Key to Calculate a Signature	84
	Guidelines and Limitations	86
	Creating a Local User and Adding a User Certificate Using the REST API	87

CHAPTER 5	Common Tenant Tasks	91
	Common Tenant Tasks	91
	Tenants Overview	91
	Tenant Creation	91
	Adding a Tenant	91
	Example: Using the JSON API to Add a Tenant	92
	Example: Using the XML API to Add a Tenant	93

CHAPTER 6	Managing Layer 2 Networking	95
	Tenant External Bridged Networks	95
	Bridged Interface to an External Router	95
	VRF and Bridge Domains	96
	Creating a Tenant, VRF, and Bridge Domain Using the REST API	96
	Ports	97
	Statically Deploying an EPG on a Specific Port	97
	Deploying an EPG on a Specific Port with APIC Using the REST API	97

Creating Domains, Attach Entity Profiles, and VLANs to Deploy an EPG on a Specific Port	98
Creating AEP, Domains, and VLANs to Deploy an EPG on a Specific Port Using the REST API	98
Creating a Port Channel Policy Using the REST API	100

CHAPTER 7**Managing Layer 3 Networking 101**

Configuring External Connectivity Using a Layer 3 Out	101
Configuring a Tenant Layer 3 Outside Network Connection Overview	101
Configuring Layer 3 Outside for Tenant Networks Using the REST API	102
Configuring BGP Max Path	105
Configuring BGP Max Path Using the REST API	105
Configuring AS Path Prepend	105
Configuring AS Path Prepend Using the REST API	106
Configuring BFD	106
Configuring BFD Globally Using the REST API	106
Configuring BFD Interface Override Using the REST API	107
Configuring BFD Consumer Protocols Using the REST API	107

CHAPTER 8**Monitoring Using the REST API 111**

About Monitoring Using the REST API	111
Monitoring APIC Using the REST API	111
APIC	111
Monitoring APIC CPU and Memory Usage Using the REST API	111
Monitoring APIC Disk Utilization Using the REST API	112
Monitoring Physical Interface Statistics and Link State Using the REST API	112
Fabric	113
Monitoring LLDP and CDP Neighbor Status Using the REST API	113
Monitoring Physical and Bond Interfaces Using the REST API	113
Monitoring EPG-Level Statistics Using the REST API	113
Switches	114
Monitoring Switch CPU Utilization Using the REST API	114
Monitoring Switch Fan Status Using the REST API	115
Monitoring Switch Memory Utilization Using the REST API	115
Monitoring Switch Module Status Using the REST API	116
Monitoring Switch Power Supply Status Using the REST API	116

Monitoring Switch Inventory Using the REST API	116
External Monitoring	117
Smart Callhome	117
About Smart Callhome	117
Creating a Smart Callhome Destination Group Using the REST API	117
TACACS External Logging	118
About TACACS External Logging	118
Creating a TACACS External Logging Destination Group Using the REST API	119
Creating a TACACS External Logging Source Using the REST API	119

CHAPTER 9**Troubleshooting Using the REST API 121**

Collecting and Exporting Technical Support Information	121
About Exporting Files	121
Sending an On-Demand Tech Support File Using the REST API	121
Troubleshooting Using Atomic Counters	122
Atomic Counters	122
Enabling Atomic Counters	123
About Fabric Latency	124
About PTP	126
Troubleshooting Using Atomic Counters with the REST API	128
Configuring Latency and PTP Using the REST API	128
Troubleshooting Using Faults	129
Understanding APIC Faults	129
Troubleshooting Using Faults with the REST API	130
Statistics	131
Configuring a Stats Monitoring Policy Using the REST API	131
Recovering a Disconnected Leaf	132
Recovering a Disconnected Leaf	132
Recovering a Disconnected Leaf Using the REST API	132
Troubleshooting Contracts and Taboo Contracts with Permit and Deny Logging	133
Verifying Contracts, Taboo Contracts, and Filters Using the REST API	133
Viewing ACL Permit and Deny Logs Using the REST API	133
Troubleshooting Using Digital Optical Monitoring Statistics	135
Troubleshooting Using Digital Optical Monitoring With the REST API	135

Troubleshooting Using Port Tracking	135
Port Tracking Policy for Fabric Port Failure Detection	135
Port Tracking Using the REST API	136
Removing Unwanted _ui_ Objects	137
Removing Unwanted _ui_ Objects Using the REST API	137
Troubleshooting Using Contract Permit and Deny Logs	137
About ACL Contract Permit and Deny Logs	137
Enabling ACL Contract Permit Logging Using the REST API	138
Enabling Taboo Contract Deny Logging Using the REST API	139
Viewing ACL Permit and Deny Logs Using the REST API	139

PART III
Part 3: Setting Up APIC and the Fabric Using the REST API 141

CHAPTER 10
Managing APIC Clusters 143

Cluster Management Guidelines	143
Cluster Management Guidelines	143
Expanding and Contracting Clusters	144
Expanding the APIC Cluster Size	144
Expanding the Cisco APIC Cluster	145
Expanding the APIC Cluster Using the REST API	145
Contracting the Cisco APIC Cluster	145
Contracting the APIC Cluster Using the REST API	146
Managing Cluster High Availability	146
About Cold Standby for a Cisco APIC Cluster	146
Switching Over Active APIC with Standby APIC Using REST API	147

CHAPTER 11
Managing Fabrics 149

Maintenance Mode	149
Removing a Switch to Maintenance Mode Using the REST API	151
Inserting a Switch to Operation Mode Using the CLI	151

CHAPTER 12
Configuring Tenant Policies 153

Basic Tenant Configuration	153
Creating a Tenant, VRF, and Bridge Domain Using the REST API	153

Tenants in Multiple Private Networks	154
About Multiple Private Networks with Inter-Tenant Communication	154
Configuring Multiple Private Networks with Inter-Tenant Communication Using the REST API	155
About Multiple Private Networks with Intra-Tenant Communication	156
Configuring Multiple Tenants with Intra-Tenant Communication Using the REST API	157
Tenant Policy Example	158
Tenant Policy Example Overview	158
Tenant Policy Example XML Code	159
Tenant Policy Example Explanation	160
Policy Universe	161
Tenant Policy Example	161
Filters	161
Contracts	162
Subjects	163
Labels	163
VRF	164
Bridge Domains	164
Application Profiles	165
Endpoints and Endpoint Groups (EPGs)	165
Closing	167
What the Example Tenant Policy Does	167
EPGs	168
Deploying an Application EPG through an AEP or Interface Policy Group to Multiple Ports	168
Deploying an EPG on a Specific Port with APIC Using the REST API	169
Deploying an EPG through an AEP to Multiple Interfaces Using the REST API	169
Creating AEP, Domains, and VLANs to Deploy an EPG on a Specific Port Using the REST API	170
Intra-EPG Isolation	171
Intra-EPG Isolation for Bare Metal Servers	171
Configuring Intra-EPG Isolation for Bare Metal Servers Using the REST API	172
Intra-EPG Isolation for VMware VDS or Microsoft Hyper-V Virtual Switch	173
Configuring Intra-EPG Isolation for VMware VDS or Microsoft Hyper-V Virtual Switch using the REST API	175
Intra-EPG Isolation Enforcement for Cisco AVS	176
Configuring Intra-EPG Isolation for Cisco AVS Using the REST API	176

Microsegmentation	177
Using Microsegmentation with Network-based Attributes on Bare Metal	177
Configuring an IP-based Microsegmented EPG as a Shared Resource Using the REST API	177
Configuring a Network-Based Microsegmented EPG in a Bare-Metal Environment Using the REST API	178
Configuring Microsegmentation on Virtual Switches	179
Configuring Microsegmentation with Cisco ACI Using the REST API	179
Application Profiles	180
Three-Tier Application Deployment	180
Parameters to Create a Filter for http	181
Parameters to Create Filters for rmi and sql	182
Deploying an Application Profile Using the REST API	182
Contracts, Taboo Contracts, and Preferred Groups	184
Security Policy Enforcement	184
Contracts and Taboo Contracts	185
Contracts Contain Security Policy Specifications	185
Contracts	187
Configuring a Contract Using the REST API	189
Configuring a Taboo Contract Using the REST API	189
Contract and Subject Exceptions	190
Configuring Contract or Subject Exceptions for Contracts	190
Configure a Contract or Subject Exception Using the REST API	191
Configuring EPG Contract Inheritance Using the REST API	191
About Contract Inheritance	191
Configuring Application EPG Contract Inheritance Using the REST API	193
Configuring uSeg EPG Contract Inheritance Using the REST API	193
Configuring L2Out EPG Contract Inheritance Using the REST API	194
Configuring L3Out EPG Contract Inheritance Using the REST API	195
Contract Preferred Groups	196
About Contract Preferred Groups	196
Configuring Contract Preferred Groups Using the REST API	198
Configuring an Enforced Bridge Domain	199
Configuring an Enforced Bridge Domain Using the REST API	200

CHAPTER 13	Provisioning Core Services	201
	DHCP	201
	Configuring a DHCP Relay Policy	201
	Configuring a DHCP Server Policy for the APIC Infrastructure Using the REST API	201
	Layer 2 and Layer 3 DHCP Relay Sample Policies	203
	DNS	205
	DNS	205
	Configuring a DNS Service Policy to Connect with DNS Providers Using the REST API	205
	DNS Policy Example	206
	NTP	207
	Time Synchronization and NTP	207
	Configuring NTP Using the REST API	207
	Tetration	208
	Overview	208
	Configuring Cisco Tetration Analytics Using the REST API	208
	NetFlow	209
	About NetFlow	209
	NetFlow on EX Platform Switches	209
	Configuring a NetFlow Exporter Policy for VM Networking Using the REST API	210
	Configuring NetFlow Infra Selectors Using REST API	210
	Configuring NetFlow Tenant Hierarchy Using REST API	211
	Consuming a NetFlow Exporter Policy Under a VMM Domain Using the REST API for VMware VDS	213
	Configuring NetFlow or Tetration Analytics Priority Using REST API	213
	DOM Statistics	214
	About Digital Optical Monitoring	214
	Enabling Digital Optical Monitoring Using the REST API	214
	Syslog	215
	About Syslog	215
	Configuring a Syslog Group and Destination Using the REST API	216
	Creating a Syslog Source Using the REST API	216
	Enabling Syslog to Display in NX-OS CLI Format, Using the REST API	217
	Data Plane Policing	218

Overview of Data Plane Policing	218
Configuring Data Plane Policing Using the REST API	219
Traffic Storm Control	220
About Traffic Storm Control	220
Configuring a Traffic Storm Control Policy Using the REST API	221
Rogue Endpoint Control	222
About the Rogue Endpoint Control Policy	222
Configure the Rogue Endpoint Control Policy Using the REST API	222

CHAPTER 14**Provisioning Layer 2 Networks 225**

Networking Domains, VLANs, and AEPs	225
Networking Domains	225
Configuring a Physical Domain Using the REST API	226
Creating VLAN Pools	226
Creating a VLAN Pool Using the REST API	227
Configuring Q-in-Q Encapsulation Mapping for EPGs	227
Q-in-Q Encapsulation Mapping for EPGs	227
Mapping EPGs to Q-in-Q Encapsulation Enabled Interfaces Using the REST API	228
Attachable Entity Profile	229
Creating an Attachable Access Entity Profile Using the REST API	230
Interfaces	231
Ports, PCs, and VPCs	231
Configuring a Single Port Channel Applied to Multiple Switches	231
Configuring a Single Virtual Port Channel Across Two Switches Using the REST API	232
Configuring Two Port Channels Applied to Multiple Switches Using the REST API	233
Configuring a Virtual Port Channel on Selected Port Blocks of Two Switches Using the REST API	234
Configuring a Virtual Port Channel and Applying it to a Static Port Using the REST API	235
Reflective Relay (802.1Qbg)	237
Enabling Reflective Relay Using the REST API	237
Interface Speed	238
Interface Configuration Guidelines	238
Changing Interface Speed	239
FEXs	240

ACI FEX Guidelines	240
Configuring an FEX VPC Policy Using the REST API	240
FCoE	243
Supporting Fibre Channel over Ethernet Traffic on the ACI Fabric	243
Configuring FCoE Connectivity Using the REST API	245
Configuring FCoE Over FEX Using REST API	248
Undeploying FCoE Connectivity through the REST API or SDK	252
Fibre Channel NPV	257
Fibre Channel Connectivity Overview	257
Fibre Channel N-Port Virtualization Guidelines and Limitations	259
Configuring FC Connectivity Using the REST API	260
802.1Q Tunnels	264
About ACI 802.1Q Tunnels	264
Configuring 802.1Q Tunnels With Ports Using the REST API	266
Configuring 802.1Q Tunnels With PCs Using the REST API	267
Configuring 802.1 Q Tunnels With vPCs Using the REST API	269
Breakout Ports	271
Configuration of Dynamic Breakout Ports	271
Configuring Dynamic Breakout Ports Using the REST API	273
Port Profiles to Change Uplinks to Downlinks and Downlinks to Uplinks	276
Configuring Port Profiles	276
Port Profile Configuration Summary	279
Configuring a Port Profile Using the REST API	280
IGMP Snooping	281
About Cisco APIC and IGMP Snooping	281
How IGMP Snooping is Implemented in the ACI Fabric	282
Virtualization Support	283
Configuring and Assigning an IGMP Snooping Policy to a Bridge Domain using the REST API	283
Enabling Group Access to IGMP Snooping and Multicast using the REST API	284
Enabling IGMP Snooping and Multicast on Static Ports Using the REST API	285
Proxy ARP	285
About Proxy ARP	285
Guidelines and Limitations	291
Configuring Proxy ARP Using the REST API	291

Flood on Encapsulation	292
Configuring Flood in Encapsulation for All Protocols and Proxy ARP Across Encapsulations	292
Configuring Flood on Encapsulation Using the REST API	297
MACsec	297
About MACsec	297
Guidelines and Limitations for MACsec	298
Configuring MACsec Using the REST API	301
<hr/>	
CHAPTER 15	Provisioning Layer 3 Outside Connections 305
Layer 3 Outside Connections	305
Configuring a Tenant Layer 3 Outside Network Connection Overview	305
Configuring Layer 3 Outside for Tenant Networks Using the REST API	306
Configuring Layer 3 Outside for Tenant Networks Using the REST API	309
REST API Example: L3Out Prerequisites	311
REST API Example: L3Out	312
REST API Example: Tenant External Network Policy	313
Layer 3 Routed and Sub-Interface Port Channels	315
About Layer 3 Port Channels	315
Configuring Port Channels Using the REST API	316
Configuring a Layer 3 Routed Port Channel Using the REST API	317
Configuring a Layer 3 Sub-Interface Port Channel Using the REST API	318
Cisco ACI GOLF	320
Cisco ACI GOLF	320
Configuring GOLF Using the REST API	321
Distributing BGP EVPN Type-2 Host Routes to a DCIG	327
Enabling Distributing BGP EVPN Type-2 Host Routes to a DCIG Using the REST API	328
Multipod	328
Multipod	328
Setting Up Multi-Pod Fabric Using the REST API	330
Anycast Services	333
About Anycast Services	333
Configuring Anycast Services Using the REST API	333
Remote Leaf Switches	335
About Remote Leaf Switches in the ACI Fabric	335

Remote Leaf Switch Hardware Requirements	339
Remote Leaf Switch Restrictions and Limitations	340
WAN Router and Remote Leaf Switch Configuration Guidelines	342
Configure Remote Leaf Switches Using the REST API	343
Prerequisites Required Prior to Downgrading Remote Leaf Switches	346
HSRP	347
About HSRP	347
Guidelines and Limitations	348
Configuring HSRP in APIC Using REST API	349
IP Multicast	351
Tenant Routed Multicast	351
Guidelines and Restrictions for Configuring Layer 3 Multicast	352
Configuring Layer 3 Multicast Using REST API	354
Pervasive Gateway	357
Common Pervasive Gateway	357
Configuring Common Pervasive Gateway Using the REST API	358
Explicit Prefix Lists	358
About Explicit Prefix List Support for Route Maps/Profile	358
Guidelines and Limitations	360
About Route Map/Profile	361
Aggregation Support for Explicit Prefix List	362
Configuring Route Map/Profile with Explicit Prefix List Using REST API	365
IP Address Aging Tracking	366
Overview	366
Configuring IP Aging Using the REST API	367
Route Summarization	367
Configuring Route Summarization for BGP, OSPF, and EIGRP Using the REST API	367
Configuring Route Summarization for BGP, OSPF, and EIGRP Using the REST API	369
Route Controls	371
About Configuring a Routing Control Protocol Using Import and Export Controls	371
Configuring a Route Control Protocol to Use Import and Export Controls, With the REST API	371
Layer 3 to Layer 3 Out Inter-VRF Leaking	372
Layer 3 Out to Layer 3 Out Inter-VRF Leaking	372
Configuring Two Shared Layer 3 Outs in Two VRFs Using REST API	373

Overview Interleak Redistribution for MP-BGP	374
Configuring Interleak of External Routes Using the REST API	374
SVI External Encapsulation Scope	375
About SVI External Encapsulation Scope	375
Encapsulation Scope Syntax	377
Configuring SVI Interface Encapsulation Scope Using the REST API	377
SVI Auto State	378
About SVI Auto State	378
Guidelines and Limitations for SVI Auto State Behavior	379
Configuring SVI Auto State Using the REST API	379
Routing Protocols	380
BGP and BFD	380
Guidelines for Configuring a BGP Layer 3 Outside Network Connection	380
BGP Connection Types and Loopback Guidelines	381
Per VRF Per Node BGP Timer Values	381
Configuring an MP-BGP Route Reflector Using the REST API	382
Configuring BGP External Routed Network Using the REST API	383
Configuring BFD Consumer Protocols Using the REST API	384
Configuring BFD Globally Using the REST API	387
Configuring BFD Interface Override Using the REST API	387
Configuring a Per VRF Per Node BGP Timer Using the REST API	388
Deleting a Per VRF Per Node BGP Timer Using the REST API	388
Configuring BGP Max Path	389
Configuring AS Path Prepend	389
About BGP Autonomous System Override	390
Configuring BGP External Routed Network with Autonomous System Override Enabled Using the REST API	391
OSPF	394
OSPF Layer 3 Outside Connections	394
Creating OSPF External Routed Network for Management Tenant Using REST API	395
EIGRP	396
Overview	396
Configuring EIGRP Using the REST API	396
Neighbor Discovery	398

Neighbor Discovery	398
Creating the Tenant, VRF, and Bridge Domain with IPv6 Neighbor Discovery on the Bridge Domain Using the REST API	400
Guidelines and Limitations	400
Configuring an IPv6 Neighbor Discovery Interface Policy with RA on a Layer 3 Interface Using the REST API	400
Microsoft NLB	402
Configuring Microsoft NLB in Unicast Mode Using the REST API	402
Configuring Microsoft NLB in Multicast Mode Using the REST API	402
Configuring Microsoft NLB in IGMP Mode Using the REST API	403
MLD Snooping	403
Configuring and Assigning an MLD Snooping Policy to a Bridge Domain using the REST API	403

CHAPTER 16**Configuring QoS 405**

QoS for L3Outs	405
L3Outs QoS	405
Configuring QoS Directly on L3Out Using REST API	405
Configuring QoS Contract for L3Out Using REST API	406
CoS Preservation	407
Class of Service (CoS) Preservation for Ingress and Egress Traffic	407
Enable Class Of Service (CoS) Preservation Using REST API	408
Multipod QoS	408
Multipod QoS and DSCP Translation Policy	408
Creating DSCP Translation Policy Using REST API	409
Translating QoS Ingress Markings to Egress Markings	410
Translating Ingress to Egress QoS Markings	410
Creating Custom QoS Policy Using REST API	410
Troubleshooting Cisco APIC QoS Policies	411

CHAPTER 17**Managing Layer 4 to Layer 7 Services 413**

About Layer 4 to Layer 7 Services	413
About Application-Centric Infrastructure Layer 4 to Layer 7 Services	413
Access for Managing Layer 4 to Layer 7 Services	414
Configure In-Band Connectivity to Devices Using Tenant's VRF Using the REST API	414

Configuring In-Band Connectivity to Devices Using Management Tenant VRF Using the REST API	415
Device Packages	417
About the Device Package	417
Notes for Installing a Device Package with the REST APIs	417
Uploading a Device Package File Using the API	418
Installing a Device Package Using the REST API	418
Using an Imported Device with the REST APIs	418
Trunking	419
About Trunking	419
Enabling Trunking on a Layer 4 to Layer 7 Virtual ASA device Using the REST APIs	419
Device Selection Policies	420
About Device Selection Policies	420
Creating a Device Selection Policy Using the REST API	420
Adding a Logical Interface in a Device Using the REST APIs	420
Policy Based Redirect and Service Nodes Tracking	421
Policy-Based Redirect and Tracking Service Nodes	421
Policy-Based Redirect and Threshold Settings for Tracking Service Nodes	421
Guidelines and Limitations for Policy-Based Redirect With Tracking Service Nodes	422
Configuring PBR to Support Tracking Service Nodes Using the REST API	423
About Location-Aware Policy Based Redirect	423
Guidelines for Location-Aware PBR	424
Configuring Location-Aware PBR Using the REST API	425
About Layer 1/Layer 2 Policy-Based Redirect	425
Guidelines and Limitations for Layer 1/Layer 2 Policy-Based Redirect	426
Configuring Layer 1/ Layer 2 PBR Using the REST API	426
Service Graph Templates	427
About Service Graph Templates	427
Configuring a Service Graph Template Using the REST APIs	428
Creating a Security Policy Using the REST APIs	429
Layer 4 to Layer 7 Parameters	429
About Modifying the Configuration Parameters of a Deployed Service Graph	429
Example XML POST for an Application EPG With Configuration Parameters	429
Example XML of Configuration Parameters Inside the Device Package	431

Example XML POST for an Abstract Function Node With Configuration Parameters	431
Example XML POST for an Abstract Function Profile With Configuration Parameters	432
Copy Services	433
About Copy Services	433
Configuring Copy Services Using the REST API	433
Developing Automation	435
About the REST APIs	435
Examples of Automating Using the REST APIs	435
Example: Configuring Layer 4 to Layer 7 Services (Firewall)	443
Example: Configuring Layer 4 to Layer 7 Services Using the REST API	443
Example: Configuring Layer 4 to Layer 7 Route Peering	452
Configuring Layer 4 to Layer 7 Route Peering With the REST API	452
Specifying an l3extOut Policy for Layer 4 to L7 Route Peering	454

CHAPTER 18**Configuring Security 457**

Enabling TACACS+, RADIUS, and LDAP	457
Overview	457
Configuring APIC for TACACS+ Using the REST API	457
Configuring APIC for RADIUS Using the REST API	458
Configuring APIC for LDAP Using the REST API	459
Configuring FIPS	460
About Federal Information Processing Standards (FIPS)	460
Guidelines and Limitations for FIPS	460
Configuring FIPS for Cisco APIC Using REST API	461
Configuring Fabric Secure Mode	461
Fabric Secure Mode	461
Configuring Fabric Secure Mode Using the REST API	462
Enabling RBAC	462
Access Rights Workflow Dependencies	462
AAA RBAC Roles and Privileges	463
Custom Roles	472
Sample RBAC Rules	473
Enabling Port Security	476
About Port Security and ACI	476

Port Security Guidelines and Restrictions	476
Port Security and Learning Behavior	476
Port Security at Port Level	477
Protect Mode	477
Configuring Port Security Using REST API	477
Enabling COOP Authentication	478
Overview	478
Using COOP with Cisco APIC	479
Guidelines and Limitations	479
Configuring COOP Authentication Using the REST API	479
Enabling Control Plane Policing	479
About Control Plane Policing	479
Guidelines and Limitations for CoPP	482
Configuring CoPP Using the REST API	483
Configuring CoPP Per Interface Per Protocol Using REST API	483
Configuring First Hop Security	484
About First Hop Security	484
ACI FHS Deployment	485
Guidelines and Limitations	485
Configuring FHS in APIC Using REST API	486
Configuring 802.1x	487
802.1X Overview	487
Host Support	487
Authentication Modes	487
Guidelines and Limitations	488
Configuration Overview	489
Configuring 802.1X Node Authentication Using the REST API	489
Configuring 802.1X Port Authentication Using the REST API	490

CHAPTER 19
Creating Quota Management 491

About APIC Quota Management Configuration	491
Creating a Quota Management Configuration Using the REST API	491

CHAPTER 20
Configuring a Forwarding Scale Profile Policy 493

[Forwarding Scale Profile Policy Overview](#) 493

[Supported Platforms](#) 495

[Guidelines and Limitations](#) 496

[Configuring the Forwarding Scale Profile Policy Using the REST API](#) 497

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
- Cloud 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.
<i>Italic</i>	Italic text indicates arguments for which the user supplies the values.
[x]	Square brackets enclose an optional element (keyword or argument).
[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.
<i>variable</i>	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
<code>screen font</code>	Terminal sessions and information the switch displays are in screen font.
<code>boldface screen font</code>	Information you must enter is in boldface screen font.

Convention	Description
<i>italic screen font</i>	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

Cisco Application Centric Infrastructure (ACI) Documentation

The ACI documentation is available at the following URL: <http://www.cisco.com/c/en/us/support/cloud-systems-management/application-policy-infrastructure-controller-apic/tsd-products-support-series-home.html>.

Cisco Application Centric Infrastructure (ACI) Simulator Documentation

The Cisco ACI Simulator documentation is available at <http://www.cisco.com/c/en/us/support/cloud-systems-management/application-centric-infrastructure-simulator/tsd-products-support-series-home.html>.

Cisco Nexus 9000 Series Switches Documentation

The Cisco Nexus 9000 Series Switches documentation is available at <http://www.cisco.com/c/en/us/support/switches/nexus-9000-series-switches/tsd-products-support-series-home.html>.

Cisco Application Virtual Switch Documentation

The Cisco Application Virtual Switch (AVS) documentation is available at <http://www.cisco.com/c/en/us/support/switches/application-virtual-switch/tsd-products-support-series-home.html>.

Cisco Application Centric Infrastructure (ACI) Integration with OpenStack Documentation

Cisco ACI integration with OpenStack documentation is available at <http://www.cisco.com/c/en/us/support/cloud-systems-management/application-policy-infrastructure-controller-apic/tsd-products-support-series-home.html>.

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.



CHAPTER 1

New and Changed Information

- [New and Changed Information](#), on page 1

New and Changed Information

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

Table 1: New and Changed Behavior in Cisco ACI, Release 4.1(1)

Feature	Description	Where Documented
L1/L2 PBR	PBR support with Layer 1 or Layer 2 service device	Managing Layer 4 to Layer 7 Services , on page 413
MLD snooping	Support for Multicast Listener Discovery (MLD) snooping	Managing Layer 3 Networking , on page 101
Microsoft NLB	Support for Microsoft Network Load Balancing (NLB)	Managing Layer 3 Networking , on page 101



PART I

Part 1: Cisco APIC REST API Usage Guidelines

- [Using the REST API, on page 5](#)



CHAPTER 2

Using the REST API

- [About the REST API, on page 5](#)
- [Composing REST API Requests, on page 8](#)
- [Composing REST API Queries, on page 18](#)
- [REST API Examples, on page 27](#)
- [Accessing the REST API, on page 35](#)
- [REST API Tools, on page 45](#)

About the REST API

The Application Policy Infrastructure Controller (APIC) REST API is a programmatic interface that uses REST architecture. The API accepts and returns HTTP (not enabled by default) or HTTPS messages that contain JavaScript Object Notation (JSON) or Extensible Markup Language (XML) documents. You can use any programming language to generate the messages and the JSON or XML documents that contain the API methods or Managed Object (MO) descriptions.

The REST API is the interface into the management information tree (MIT) and allows manipulation of the object model state. The same REST interface is used by the APIC CLI, GUI, and SDK, so that whenever information is displayed, it is read through the REST API, and when configuration changes are made, they are written through the REST API. The REST API also provides an interface through which other information can be retrieved, including statistics, faults, and audit events. It even provides a means of subscribing to push-based event notification, so that when a change occurs in the MIT, an event can be sent through a web socket.

Standard REST methods are supported on the API, which includes POST, GET, and DELETE operations through HTTP. The POST and DELETE methods are idempotent, meaning that there is no additional effect if they are called more than once with the same input parameters. The GET method is nullipotent, meaning that it can be called zero or more times without making any changes (or that it is a read-only operation).

Payloads to and from the REST interface can be encapsulated through either XML or JSON encoding. In the case of XML, the encoding operation is simple: the element tag is the name of the package and class, and any properties of that object are specified as attributes of that element. Containment is defined by creating child elements.

For JSON, encoding requires definition of certain entities to reflect the tree-based hierarchy; however, the definition is repeated at all levels of the tree, so it is fairly simple to implement after it is initially understood.

- All objects are described as JSON dictionaries, in which the key is the name of the package and class. The value is another nested dictionary with two keys: attribute and children.

- The attribute key contains a further nested dictionary describing key-value pairs that define attributes on the object.
- The children key contains a list that defines all the child objects. The children in this list are dictionaries containing any nested objects, which are defined as described here.

Authentication

REST API username- and password-based authentication uses a special subset of request Universal Resource Identifiers (URIs), including **aaaLogin**, **aaaLogout**, and **aaaRefresh** as the DN targets of a POST operation. Their payloads contain a simple XML or JSON payload containing the MO representation of an **aaaUser** object with the attribute name and **pwd** defining the username and password: for example, `<aaaUser name='admin' pwd='password'/>`. The response to the POST operation will contain an authentication token as both a Set-Cookie header and an attribute to the **aaaLogin** object in the response named token, for which the XPath is `/imdata/aaaLogin/@token` if the encoding is XML. Subsequent operations on the REST API can use this token value as a cookie named **APIC-cookie** to authenticate future requests.

Subscription

The REST API supports the subscription to one or more MOs during your active API session. When any MO is created, changed, or deleted because of a user- or system-initiated action, an event is generated. If the event changes the data on any of the active subscribed queries, the APIC will send out a notification to the API client that created the subscription.

Management Information Model

All the physical and logical components that comprise the Application Centric Infrastructure fabric are represented in a hierarchical management information model (MIM), also referred to as the MIT. Each node in the tree represents an MO or group of objects that contains its administrative state and its operational state.

To view the MIM, see *Cisco APIC Management Information Model Reference Guide*.

The hierarchical structure starts at the top (Root) and contains parent and child nodes. Each node in this tree is an MO and each object in the ACI fabric has a unique distinguished name (DN) that describes the object and its place in the tree. MOs are abstractions of the fabric resources. An MO can represent a physical object, such as a switch or adapter, or a logical object, such as a policy or fault.

Configuration policies make up the majority of the policies in the system and describe the configurations of different ACI fabric components. Policies determine how the system behaves under specific circumstances. Certain MOs are not created by users but are automatically created by the fabric (for example, power supply objects and fan objects). By invoking the API, you are reading and writing objects to the MIM.

The information model is centrally stored as a logical model by the APIC, while each switch node contains a complete copy as a concrete model. When a user creates a policy in the APIC that represents a configuration, the APIC updates the logical model. The APIC then performs the intermediate step of creating a fully elaborated policy from the user policy and then pushes the policy into all the switch nodes where the concrete model is updated. The models are managed by multiple data management engine (DME) processes that run in the fabric. When a user or process initiates an administrative change to a fabric component (for example, when you apply a profile to a switch), the DME first applies that change to the information model and then applies the change to the actual managed endpoint. This approach is called a model-driven framework.

The following branch diagram of a leaf switch port starts at the top Root of the ACI fabric MIT and shows a hierarchy that comprises a chassis with two line module slots, with a line module in slot 2.


```

|--root----- (root)
  |--sys----- (sys)
    |--ch----- (sys/ch)
      |--lcslot-1----- (sys/ch/lcslot-1)
      |--lcslot-2----- (sys/ch/lcslot-2)
        |--lc----- (sys/ch/lcslot-2/lc)
          |--leafport-1----- (sys/ch/lcslot-2/lc/leafport-1)

```

Object Naming

You can identify a specific object by its distinguished name (DN) or by its relative name (RN).



Note You cannot rename an existing object. To simplify references to an object or group of objects, you can assign an alias or a tag.

Distinguished Name

The DN enables you to unambiguously identify a specific target object. The DN consists of a series of RNs:

```
dn = {rn}/{rn}/{rn}/{rn}...
```

In this example, the DN provides a fully qualified path for `fabport-1` from the top of the object tree to the object. The DN specifies the exact managed object on which the API call is operating.

```
< dn = "sys/ch/lcslot-1/lc/fabport-1" />
```

Relative Name

The RN identifies an object from its siblings within the context of its parent object. The DN contains a sequence of RNs.

For example, this DN:

```
<dn = "sys/ch/lcslot-1/lc/fabport-1"/>
```

contains these RNs:

Relative Name	Class	Description
sys	top:System	Top level of this system
ch	eqpt:Ch	Hardware chassis container
lcslot-1	eqpt:LCSlot	Line module slot 1
lc	eqpt:LC	Line (I/O) module
fabport-1	eqpt:FabP	Fabric-facing external I/O port 1

Guidelines and Limitations for Using the REST API

The following guidelines and limitations apply when using the Cisco Application Policy Infrastructure Controller (APIC) REST API:

- On scale setups, if you send generic class queries to the Cisco APIC that result in a large set of managed objects, the queries intermittently fail due to a timeout with error code 503 and the following error message:

```
Unable to deliver the message, destination is not available  
Unable to process the query, result dataset is too big
```

For REST API queries on a class that has more than 100,000 objects across the fabric, the Cisco APIC generates the indicated errors due to one of the following reasons:

- Cisco APIC does not respond with more than 100,000 objects to avoid an out-of-memory issue. The APIC returns the "too big" error.
- Cisco APIC allows a maximum of 90 seconds to respond to any query that possibly timed out due to having too many activities. In this case, the Cisco APIC responds with "destination not available" because the destination could not finish the request in 90 seconds.

To mitigate this limitation:

- On a timeout response, such as "destinations not available," have the client retry from 3 to 5 times.
- If the response is the "too big" error, the client can use filtering to reduce the size of the result set.
- If the system page indicates that there are too many critical faults, we recommend that you take care of the faults.

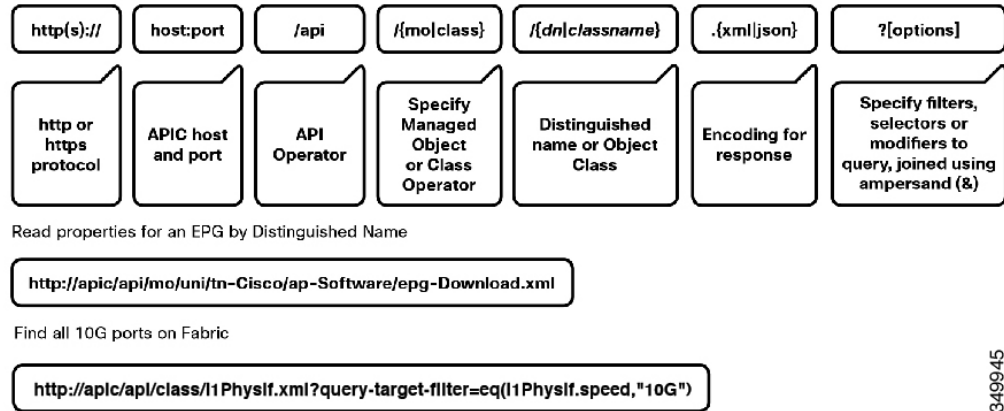
Composing REST API Requests

Read and Write Operations and Filters

Read Operations

After the object payloads are properly encoded as XML or JSON, they can be used in create, read, update, or delete operations on the REST API. The following diagram shows the syntax for a read operation from the REST API.

Figure 1: REST syntax



Because the REST API is HTTP-based, defining the URI to access a certain resource type is important. The first two sections of the request URI simply define the protocol and access details of the APIC. Next in the request URI is the literal string `/api`, indicating that the API will be invoked. Generally, read operations are for an object or class, as discussed earlier, so the next part of the URI specifies whether the operation will be for an MO or class. The next component defines either the fully qualified domain name (DN) being queried for object-based queries, or the package and class name for class-based queries. The final mandatory part of the request URI is the encoding format: either `.xml` or `.json`. This is the only method by which the payload format is defined. (The APIC ignores Content-Type and other headers.)

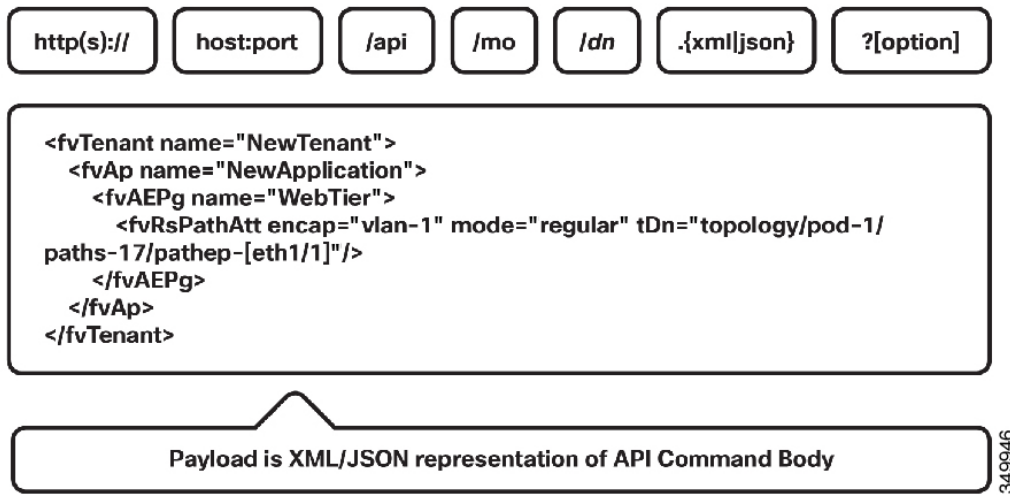
Write Operations

Both create and update operations in the REST API are implemented using the POST method, so that if an object does not already exist, it will be created, and if it does already exist, it will be updated to reflect any changes between its existing state and desired state.

Both create and update operations can contain complex object hierarchies, so that a complete tree can be defined in a single command so long as all objects are within the same context root and are under the 1MB limit for data payloads for the REST API. This limit is in place to guarantee performance and protect the system under high loads.

The context root helps define a method by which the APIC distributes information to multiple controllers and helps ensure consistency. For the most part, the configuration should be transparent to the user, though very large configurations may need to be broken into smaller pieces if they result in a distributed transaction.

Figure 2: REST Payload



Create and update operations use the same syntax as read operations, except that they are always targeted at an object level, because you cannot make changes to every object of a specific class (nor would you want to). The create or update operation should target a specific managed object, so the literal string `/mo` indicates that the DN of the managed object will be provided, followed next by the actual DN. Filter strings can be applied to POST operations; if you want to retrieve the results of your POST operation in the response, for example, you can pass the `rsp-subtree=modified` query string to indicate that you want the response to include any objects that have been modified by your POST operation.

The payload of the POST operation will contain the XML or JSON encoded data representing the MO that defines the Cisco API command body.

Filters



Note For a Cisco APIC REST API query of event records, the Cisco APIC system limits the response to a maximum of 500,000 event records. If the response is more than 500,000 events, it returns an error. Use filters to refine your queries. For more information, see *Composing Query Filter Expressions* in the [Cisco APIC REST API User Guide](#).

The REST API supports a wide range of flexible filters, useful for narrowing the scope of your search to allow information to be located more quickly. The filters themselves are appended as query URI options, starting with a question mark (?) and concatenated with an ampersand (&). Multiple conditions can be joined together to form complex filters.

The following query filters are available:

Table 2: Query Filters

Filter Type	Syntax	Cobra Query Property	Description
query-target	{self children subtree}	AbstractQuery.queryTarget	Define the scope of a query

Filter Type	Syntax	Cobra Query Property	Description
target-subtree-class	<i>class name</i>	AbstractQuery.classFilter	Respond-only elements including the specified class
query-target-filter	<i>filter expressions</i>	AbstractQuery.propFilter	Respond-only elements matching conditions
rsp-subtree	{no children full}	AbstractQuery.subtree	Specifies child object level included in the response
rsp-subtree-class	<i>class name</i>	AbstractQuery.subtreeClassFilter	Respond only specified classes
rsp-subtree-filter	<i>filter expressions</i>	AbstractQuery.subtreePropFilter	Respond only classes matching conditions
rsp-subtree-include	{faults health :stats :...}	AbstractQuery.subtreeInclude	Request additional objects
order-by	<i>classname.property</i> {asc desc}	Not Implemented	Sort the response based on the property values

Using Classes in REST API Commands

The Application Policy Infrastructure Controller (APIC) classes are crucial from an operational perspective to understand how system events and faults relate to objects within the object model. Each event and/or fault in the system is a unique object that can be accessed for configuration, health, fault, and/or statistics.

All the physical and logical components that make up the Cisco Application Centric Infrastructure (ACI) fabric are represented in a hierarchical management information tree (MIT). Each node in the tree represents a managed object (MO) or group of objects that contains its administrative state and its operational state.

To access the complete list of classes, point to the APIC and reference the `doc/html` directory at the end of the URL:

```
https://apic-ip-address/doc/html/
```

Using Managed Objects in REST API Commands

Before performing an API operation on a managed object (MO) or its properties, you should view the object's class definition in the *Cisco APIC Management Information Model Reference*, which is a web-based document. The Management Information Model (MIM) serves as a schema that defines rules such as the following:

- The classes of parent objects to which an MO can be attached
- The classes of child objects that can be attached to an MO
- The number of child objects of a class type that can be attached to an MO
- Whether a user can create, modify, or delete an MO, and the privilege level required to do so
- The properties (attributes) of an object class

- The data type and range of a property

When you send an API command, the APIC checks the command for conformance with the MIM schema. If an API command violates the MIM schema, the APIC rejects the command and returns an error message. For example, you can create an MO only if it is allowed in the path you have specified in the command URI and only if you have the required privilege level for that object class. You can configure an MO's properties only with valid data, and you cannot create properties.

When composing an API command to create an MO, you need only include enough information in the command's URI and data structure to uniquely define the new MO. If you omit the configuration of a property when creating the MO, the property is populated with a default value if the MIM specifies one, or it is left blank.

When modifying a property of an MO, you need only specify the property to be modified and its new value. Other properties will be left unchanged.

Guidelines and Restrictions

- When you modify an MO that affects APIC or switch management communication policy, you might experience a brief disruption of any operations in progress on any APIC or switch web interface in the fabric. Configuration changes that can result in disruption include the following:
 - Changing management port settings, such as port number
 - Enabling or disabling HTTPS
 - Changing the state of redirection to HTTPS
 - Public key infrastructure (PKI) changes, such as key ring
- When you read an existing MO, any password property of the MO is read as blank for security reasons. If you then write the MO back to APIC, the password property is written as blank.



Tip If you need to store an MO with its password information, use a configuration export policy. To store a specific MO, specify the MO as the target distinguished name in the policy.

Creating the API Command

You can invoke an API command or query by sending an HTTP or HTTPS message to the APIC with a URI of this form for an operation on a managed object (MO):

```
{http | https}://host[:port]/api/mo/dn. {json | xml}[?options]
```

Use this form for an operation on an object class:

```
{http | https}://host[:port]/api/class/className. {json | xml}[?options]
```



Note While the preceding examples use `/api/mo` and `/api/class` in the URI string, the APIC UI and Visore also use the `/api/node/mo` and `/api/node/class` syntax in the URI string. Both formats are valid and are used interchangeably in this document.

This example shows a URI for an API operation that involves an MO of class `fv:Tenant`:

```
https://apic-ip-address/api/mo/uni/tn-ExampleCorp.xml
```

URI Components

The components of the URI are as follows:

- `http://` or `https://`—Specifies HTTP or HTTPS. By default, only HTTPS is enabled. HTTP or HTTP-to-HTTPS redirection, if desired, must be explicitly enabled and configured, as described in [Configuring HTTP and HTTPS Using the GUI, on page 36](#). HTTP and HTTPS can coexist.
- `host`—Specifies the hostname or IP address of the APIC.
- `:port`—Specifies the port number for communicating with the APIC. If your system uses standard port numbers for HTTP (80) or HTTPS (443), you can omit this component.
- `/api/`—Specifies that the message is directed to the API.
- `mo | class`—Specifies whether the target of the operation is an MO or an object class.
- `dn`—Specifies the distinguished name (DN) of the targeted MO.
- `className`—Specifies the name of the targeted class. This name is a concatenation of the package name of the object queried and the name of the class queried in the context of the corresponding package. For example, the class `aaa:User` results in a `className` of `aaaUser` in the URI.
- `json | xml`—Specifies whether the encoding format of the command or response HTML body is JSON or XML.
- `?options`—(Optional) Specifies one or more filters, selectors, or modifiers to a query. Multiple option statements are joined by an ampersand (&).

The URI for an API Operation on an MO

In an API operation to create, read, update, or delete a specific MO, the resource path consists of `/mo/` followed by the DN of the MO as described in the *Cisco APIC Management Information Model Reference*. For example, the DN of a tenant object, as described in the reference definition of class `fv:Tenant`, is `uni/tn-[name]`. This URI specifies an operation on an `fv:Tenant` object named `ExampleCorp`:

```
https://apic-ip-address/api/mo/uni/tn-ExampleCorp.xml
```

Alternatively, in a POST operation, you can POST to `/api/mo` and provide the DN in the body of the message, as in this example:

```
POST https://apic-ip-address/api/mo.xml
```

```
<fvTenant dn="uni/tn-ExampleCorp"/>
```

You can also provide only the name in the message body and POST to `/api/mo` and the remaining RN components, as in this example:

```
POST https://apic-ip-address/api/mo/uni.xml
<fvTenant name="ExampleCorp"/>
```

The URI for an API Operation on a Node MO

In an API operation to access an MO on a specific node device in the fabric, the resource path consists of `/mo/topology/pod-number/node-number/sys/` followed by the node component. For example, to access a board sensor in chassis slot b of node-1 in pod-1, use this URI:

```
GET https://apic-ip-address/api/mo/topology/pod-1/node-1/sys/ch/bslot/board/sensor-3.json
```

The URI for an API Operation on a Class

In an API operation to get information about a class, the resource path consists of `/class/` followed by the name of the class as described in the *Cisco APIC Management Information Model Reference*. In the URI, the colon in the class name is removed. For example, this URI specifies a query on the class `aaa>User`:

```
GET https://apic-ip-address/api/class/aaaUser.json
```

Composing the API Command Body

The HTML body of a POST operation must contain a JSON or XML data structure that provides the essential information necessary to execute the command. No data structure is sent with a GET or DELETE operation.

Guidelines for Composing the API Command Body

- The data structure does not need to represent the entire set of attributes and elements of the target MO or method, but it must provide at least the minimum set of properties or parameters necessary to identify the MO and to execute the command, not including properties or parameters that are incorporated into the URI.
- The data structure is a single tree in which all child nodes are unique with a unique DN. Duplicate nodes are not allowed. You cannot make two changes to a node by including the same node twice. In this case, you must merge your changes into a single node.
- In the data structure, the colon after the package name is omitted from class names and method names. For example, in the data structure for an MO of class `zzz:Object`, label the class element as `zzzObject`.
- Although the JSON specification allows unordered elements, the APIC REST API requires that the JSON 'attributes' element precede the 'children' array or other elements.
- If an XML data structure contains no children or subtrees, the object element can be self-closing.
- The API is case sensitive.

- When sending an API command, with 'api' in the URL, the maximum size of the HTML body for the API POST command is 1 MB.
- When uploading a device package file, with 'ppi' in the URL, the maximum size of the HTML body for the POST command is 10 MB.

Composing the API Command Body to Call a Method

To compose a command to call a method, create a JSON or XML data structure containing the parameters of the method using the method description in the *Cisco APIC Management Information Model Reference*.

The API reference for a typical method lists its input parameters, if any, and its return values, if any. The method is called with a structure containing the essential input parameters, and a successful response returns a complete structure containing the return values.

The description for a hypothetical method config:Method might appear in the API reference as follows:

```
Method config:Method(  
    inParameter1,  
    inParameter2,  
    inParameter3,  
    outParameter1,  
    outParameter2  
)
```

The parameters beginning with "in" represent the input parameters. The parameters beginning with "out" represent values returned by the method. Parameters with no "in" or "out" prefix are input parameters.

A JSON structure to call the method resembles the following structure:

```
{  
  "configMethod":  
  {  
    "attributes":  
    {  
      "inParameter1": "value1",  
      "inParameter2": "value2",  
      "inParameter3": "value3"  
    }  
  }  
}
```

An XML structure to call the method resembles the following structure:

```
<configMethod  
  inParameter1="value1"  
  inParameter2="value2"  
  inParameter3="value3"  
>
```



Note The parameters of some methods include a substructure, such as filter settings or configuration settings for an MO. For specific information, see the method description in the *Cisco APIC Management Information Model Reference*.

Composing the API Command Body for an API Operation on an MO

To compose a command to create, modify, or delete an MO, create a JSON or XML data structure that describes the essential properties and children of the object's class by using the class description in the *Cisco APIC Management Information Model Reference*. You can omit any attributes or children that are not essential to execute the command.

A JSON structure for an MO of hypothetical class `zzz:Object` resembles this structure:

```
{
  "zzzObject" : {
    "attributes" : {
      "property1" : "value1",
      "property2" : "value2",
      "property3" : "value3"
    },
    "children" :
    [
      {
        "zzzChild1" : {
          "attributes" : {
            "childProperty1" : "childValue1",
            "childProperty2" : "childValue1"
          },
          "children" : []
        }
      }
    ]
  }
}
```

An XML structure for an MO of hypothetical class `zzz:Object` resembles this structure:

```
<zzzObject
  property1 = "value1",
  property2 = "value2",
  property3 = "value3">
  <zzzChild1
    childProperty1 = "childValue1",
    childProperty2 = "childValue1">
  </zzzChild1>
</zzzObject>
```

A successful operation returns a complete data structure for the MO.

Using Tags and Alias

To simplify API operations, you can assign tags or an alias to an object. In an API operation, you can refer to the object or group of objects by the alias or tag name instead of by the distinguished name (DN). Tags and aliases differ in their usage as follows:

- **Tag**—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.
- **Alias**—An alias can be a simpler and more descriptive name than the DN when referring to a single object. You can assign a particular alias name to only one object. The system will prevent you from assigning the same alias name to a second object.



Note Not every object supports a tag. To determine whether an object is taggable, inspect the class of the object in the *Cisco APIC Management Information Model Reference*. If the contained hierarchy of the object class includes a tag instance (such as `tag:AInst` or a class that derives from `tag:AInst`), an object of that class can be tagged.

Adding Tags

You can add one or more tags by using the following syntax in the URI of an API POST operation:

```
/api/tag/mo/dn . { json | xml } ? add = [ name, ... ] [ name, ... ]
```

In this syntax, *name* is the name of a tag and *dn* is the distinguished name of the object to which the tag is assigned.

This example shows how to assign the tags tenants and orgs to the tenant named ExampleCorp:

```
POST https://apic-ip-address/api/tag/mo/uni/tn-ExampleCorp.xml?add=tenants,orgs
```

Removing Tags

You can remove one or more tags by using the following syntax in the URI of an API POST operation:

```
/api/tag/mo/dn . { json | xml } ? remove = name [ name, ... ]
```

This example shows how to remove the tag orgs from the tenant named ExampleCorp:

```
POST https://apic-ip-address/api/tag/mo/uni/tn-ExampleCorp.xml?remove=orgs
```

You can delete all instances of a tag by using the following syntax in the URI of an API DELETE operation:

```
/api/tag/name . { json | xml }
```

This example shows how to remove the tag orgs from all objects:

```
DELETE https://apic-ip-address/api/tag/orgs.xml
```

Adding an Alias

You can add an alias by using the following syntax in the URI of an API POST operation:

```
/api/alias/mo/dn . { json | xml } ? set = name
```

In this syntax, *name* is the name of the alias and *dn* is the distinguished name of the object to which the alias is assigned.

This example shows how to assign the alias tenant8 to the tenant named ExampleCorp:

```
POST https://apic-ip-address/api/alias/mo/uni/tn-ExampleCorp.xml?set=tenant8
```

Removing an Alias

You can remove an alias by using the following syntax in the URI of an API POST operation:

```
/api /alias /mo / dn . { json | xml } ? clear = yes
```

This example shows how to remove any alias from the tenant named ExampleCorp:

```
POST https://apic-ip-address/api/alias/mo/uni/tn-ExampleCorp.xml?clear=yes
```

Additional Examples



Note In the examples in this section, the responses have been edited to remove attributes unrelated to tags.

This example shows how to find all tags assigned to the tenant named ExampleCorp:

```
GET https://apic-ip-address/api/tag/mo/uni/tn-ExampleCorp.xml
```

```
RESPONSE:
<imdata>
  <tagInst
    dn="uni/tn-ExampleCorp/tag-tenants"
    name="tenants"
  />
  <tagInst
    dn="uni/tn-ExampleCorp/tag-orgs"
    name="orgs"
  />
</imdata>
```

This example shows how to find all objects with the tag 'tenants':

```
GET https://apic-ip-address/api/tag/tenants.xml
```

```
RESPONSE:
<imdata>
  <fvTenant
    dn="uni/tn-ExampleCorp"
    name="ExampleCorp"
  />
</imdata>
```

Composing REST API Queries

Composing Query Filter Expressions

You can filter the response to an API query by applying an expression of logical operators and values.

A basic equality or inequality test is expressed as follows:

```
query-target-filter=[eq|ne] (attribute,value)
```

You can create a more complex test by combining operators and conditions using parentheses and commas:

```
query-target-filter=[and|or] ([eq|ne] (attribute,value) , [eq|ne] (attribute,value) , ...)
```



Note A scoping filter can contain a maximum of 20 '(attribute,value)' filter expressions. If the limit is exceeded, the API returns an error.

Available Logical Operators

This table lists the available logical operators for query filter expressions.

Operator	Description
eq	Equal to
ne	Not equal to
lt	Less than
gt	Greater than
le	Less than or equal to
ge	Greater than or equal to
bw	Between
not	Logical inverse
and	Logical AND
or	Logical OR
xor	Logical exclusive OR
true	Boolean TRUE
false	Boolean FALSE
anybit	TRUE if at least one bit is set
allbits	TRUE if all bits are set
wcard	Wildcard
pholder	Property holder
passive	Passive holder

Examples

This example returns all managed objects of class `aaaUser` whose last name is equal to "Washington":

```
GET https://apic-ip-address/api/class/aaaUser.json?
    query-target-filter=eq(aaaUser.lastName,"Washington")
```

This example returns endpoint groups whose `fabEncap` property is "vxlan-12780288":

```
GET https://apic-ip-address/api/class/fvAEPg.xml?
    query-target-filter=eq(fvAEPg.fabEncap,"vxlan-12780288")
```

This example shows all tenant objects with a current health score of less than 50:

```
GET https://apic-ip-address/api/class/fvTenant.json?
    rsp-subtree-include=health,required
    &
    rsp-subtree-filter=lt(healthInst.cur,"50")
```

This example returns all endpoint groups and their faults under the tenant `ExampleCorp`:

```
GET https://apic-ip-address/api/mo/uni/tn-ExampleCorp.xml?
    query-target=subtree
    &
    target-subtree-class=fvAEPg
    &
    rsp-subtree-include=faults
```

This example returns `aaa:Domain` objects whose names are not "infra" or "common":

```
GET https://apic-ip-address/api/class/aaaDomain.json?
    query-target-filter=
    and(ne(aaaDomain.name,"infra"),
    ne(aaaDomain.name,"common"))
```

Applying Query Scoping Filters

You can limit the scope of the response to an API query by applying scoping filters. You can limit the scope to the first level of an object or to one or more of its subtrees or children based on the class, properties, categories, or qualification by a logical filter expression.

query-target={self | children | subtree}

This statement restricts the scope of the query. This list describes the available scopes:

- **self** —(Default) Considers only the MO itself, not the children or subtrees.
- **children** —Considers only the children of the MO, not the MO itself.
- **subtree** —Considers the MO itself and its subtrees.

target-subtree-class=*mo-class1*[,*mo-class2*]...

This statement specifies which object classes are to be considered when the **query-target** option is used with a scope other than **self**. You can specify multiple desired object types as a comma-separated list with no spaces.

To request subtree information, combine **query-target=subtree** with the **target-subtree-class** statement to indicate the specific subtree as follows:

```
query-target=subtree&target-subtree-class=className
```

This example requests information about the running firmware. The information is contained in the `firmware:CtrlrRunning` subtree (child) object of the APIC firmware status container `firmware:CtrlrFwStatusCont`:

```
GET https://apic-ip-address/api/class/firmwareCtrlrFwStatusCont.json?
    query-target=subtree&target-subtree-class=firmwareCtrlrRunning
```

query-target-filter=*filter-expression*

This statement specifies a logical filter to be applied to the response. This statement can be used by itself or applied after the **query-target** statement.

rsp-subtree={*no* | *children* | *full*}

For objects returned, this option specifies whether child objects are included in the response. This list describes the available choices:

- **no** —(Default) Response includes no children.
- **children** —Response includes only the children.
- **full** —Response includes the entire structure including the children.

rsp-subtree-class=*mo-class*

When child objects are to be returned, this statement specifies that only child objects of the specified object class are included in the response.

rsp-subtree-filter=*filter-expression*

When child objects are to be returned, this statement specifies a logical filter to be applied to the child objects.



Note When an **rsp-subtree-filter** query statement includes a *class.property* operand, the specified class name is used only to identify the property and its type. The returned results are not filtered by class, and may include any child object that contains a property of the same name but belonging to a different class if that object's property matches the query condition. To filter by class, you must use additional query filters.

rsp-subtree-include=category1[,category2...][option]

When child objects are to be returned, this statement specifies additional contained objects or options to be included in the response. You can specify one or more of the following categories in a comma-separated list with no spaces:

- **audit-logs** —Response includes subtrees with the history of user modifications to managed objects.
- **event-logs** —Response includes subtrees with event history information.
- **faults** —Response includes subtrees with currently active faults.
- **fault-records** —Response includes subtrees with fault history information.
- **health** —Response includes subtrees with current health information.
- **health-records** —Response includes subtrees with health history information.
- **relations** —Response includes relations-related subtree information.
- **stats** —Response includes statistics-related subtree information.
- **tasks** —Response includes task-related subtree information.

With any of the preceding categories, you can also specify one of the following options to further refine the query results:

- **count** —Response includes a count of matching subtrees but not the subtrees themselves.
- **no-scoped** —Response includes only the requested subtree information. Other top-level information of the target MO is not included in the response.
- **required** —Response includes only the managed objects that have subtrees matching the specified category.

For example, to include fault-related subtrees, specify **faults** in the list. To return only fault-related subtrees and no other top-level MO information, specify **faults,no-scoped** in the list as shown in this example:

```
query-target=subtree&rsp-subtree-include=faults,no-scoped
```



Note Some types of child objects are not created until the parent object has been pushed to a fabric node (leaf). Until such a parent object has been pushed to a fabric node, a query on the parent object using the **rsp-subtree-include** filter might return no results. For example, a class query for `fVAAEPg` that includes the query option `rsp-subtree-include=stats` will return stats only for endpoint groups that have been applied to a tenant and pushed to a fabric node.

rsp-prop-include={all | naming-only | config-only}

This statement specifies what type of properties should be included in the response when the **rsp-subtree** option is used with an argument other than **no**.

- **all** —Response includes all properties of the returned managed objects.
- **naming-only** —Response includes only the naming properties of the returned managed objects.

- **config-only** —Response includes only the configurable properties of the returned managed objects.



Note If the managed object is not configurable or cannot be exported (backed up), the managed object is not returned.

Related Topics

[Composing Query Filter Expressions](#), on page 18

[Example: Using the JSON API to Get Running Firmware](#), on page 31

Filtering API Query Results

You can filter the results of an API query by appending one or more condition statements to the query URI as a parameter in this format:

```
https://URI?condition[&condition[&...]]
```

Multiple condition statements are joined by an ampersand (&).



Note The condition statement must not contain spaces.

Options are available to filter by object attributes and object subtrees.

Filter Conditional Operators

The query filtering feature supports the following condition operators:

Operator	Description
eq	Equal to
ne	Not equal to
lt	Less than
gt	Greater than
le	Less than or equal to
ge	Greater than or equal to
bw	Between
not	Logical inverse
and	Logical AND
or	Logical OR

Operator	Description
xor	Logical exclusive OR
true	Boolean TRUE
false	Boolean FALSE
anybit	TRUE if at least one bit is set
allbits	TRUE if all bits are set
wcard	Wildcard
pholder	Property holder
passive	Passive holder

Sorting and Paginating Query Results

When sending an API query that returns a large quantity of data, you can have the return data sorted and paginated to make it easier to find the information you need.

Sorting the Results

By adding the `order-by` operator to the query URI, you can sort the query response by one or more properties of a class, and you can specify the direction of the order using the following syntax.

order-by = *classname . property* [| { **asc** | **desc** }] [, *classname . property* [| { **asc** | **desc** }]] [, ...]

Use the optional pipe delimiter ("|") to specify either ascending order (`asc`) or descending order (`desc`). If no order is specified, the default is ascending order.

You can perform a multi-level sort by more than one property (for example, last name and first name), but all properties must be of the same MO or they must be inherited from the same abstract class.

This example shows you how to sort users by last name, then by first name:

```
GET
https://apic-ip-address/api/class/aaaUser.json?order-by=aaaUser.lastName|asc,aaaUser.firstName|asc
```

Paginating the Results

By adding the `page-size` operator to the query URI, you can divide the query results into groups (pages) of objects using the following syntax. The operand specifies the number of objects in each group.

page-size = *number-of-objects-per-page*

By adding the `page` operator in the query URI, you can specify a single group to be returned using the following syntax. The pages start from number 0.

page = *page-number*

This example shows you how to specify 15 fault instances per page in descending order, returning only the first page:

```
GET
https://apic-ip-address/api/class/faultInfo.json?order-by=faultInst.severity|desc&page=0&page-size=15
```



Note Every query, whether paged or not, generates a new set of results. When you perform a query that returns only a single page, the query response includes a count of the total results, but the unsent pages are not stored and cannot be retrieved by a subsequent query. A subsequent query generates a new set of results and returns the page requested in that query.

Subscribing to Query Results

When you perform an API query, you have the option to create a subscription to any future changes in the results of that query that occur during your active API session. When any MO is created, changed, or deleted because of a user- or system-initiated action, an event is generated. If that event changes the results of an active subscribed query, the APIC generates a push notification to the API client that created the subscription.

Opening a WebSocket

The API subscription feature uses the WebSocket protocol (RFC 6455) to implement a two-way connection with the API client through which the API can send unsolicited notification messages to the client. To establish this notification channel, you must first open a WebSocket connection with the API. Only a single WebSocket connection is needed to support multiple query subscriptions with multiple APIC instances. The WebSocket connection is dependent on your API session connection, and closes when your API session ends.



Note When MO events go through the event manager (`eventmgr`), clients receive notification of WebSocket subscription for any MO. Although most of the APIC MOs do go through `eventmgr`, stats objects do not go through it, because updates are very frequent and not scalable. Therefore, if you subscribe to stats objects, you will receive no notification. Instead you can periodically query or export stats MOs.

The WebSocket connection is typically opened by a JavaScript method in an HTML5-compliant browser, as in the following example:

```
var Socket = new WebSocket (https://apic-ip-address/socket%TOKEN%);
```

In the URI, the `%TOKEN%` is the current API session token (cookie). This example shows the URI with a token:

```
https://apic-ip-address/socketGkZ15NLRZJ15+jqChouaZ9CYjgE58W/pMccR+LeXmd00obG9NB
Iwo1VBo7+YC1oiJL9mS6I9qh62BkX+Xddhe0JYrTmSG4JcKZ4t3bcP2Mxy3VBmgoJjwz76ZOuf9V9AD6X
1831yoR4bLBzqbSSU1R2NIgUotCGWjZt5JX6CJF0=
```

After the WebSocket connection is established, it is not necessary to resend the API session token when the API session is refreshed.

Creating a Subscription

To create a subscription to a query, perform the query with the option `?subscription=yes`. This example creates a subscription to a query of the `fv:Tenant` class in the JSON format:

```
GET https://apic-ip-address/api/class/fvTenant.json?subscription=yes
```

To specify a refresh-timeout in seconds, perform the query with the option `?&subscription=yes&refresh-timeout=timeout-time`, where *timeout-time* is the refresh-timeout in seconds. This example creates a subscription to a query of the `fv:Tenant` class in the JSON format, with a refresh-timeout of 140 seconds:

```
GET https://apic-ip-address/api/class/fvTenant.json?&subscription=yes&refresh-timeout=140
```

The query response contains a subscription identifier, **subscriptionId**, that you can use to refresh the subscription and to identify future notifications from this subscription.

```
{
  "subscriptionId" : "72057611234574337",
  "imdata" : [{
    "fvTenant" : {
      "attributes" : {
        "instanceId" : "0:0",
        "childAction" : "",
        "dn" : "uni/tn-common",
        "lcOwn" : "local",
        "monPolDn" : "",
        "name" : "common",
        "replTs" : "never",
        "status" : ""
      }
    }
  ]
}
```

Receiving Notifications

An event notification from the subscription delivers a data structure that contains the subscription ID and the MO description. In this JSON example, a new user has been created with the name "sysadmin5":

```
{
  "subscriptionId" : ["72057598349672454", "72057598349672456"],
  "imdata" : [{
    "aaaUser" : {
      "attributes" : {
        "accountStatus" : "active",
        "childAction" : "",
        "clearPwdHistory" : "no",
        "descr" : "",
        "dn" : "uni/userext/user-sysadmin5",
        "email" : "",
        "encPwd" : "TUxISkhH$VHyidGgBX0r7N/srt/YcMYTEn5248ommFhNFzZghvAU=",
        "expiration" : "never",
        "expires" : "no",
        "firstName" : "",
        "intId" : "none",

```

```

        "lastName" : "",
        "lcOwn" : "local",
        "name" : "sysadmin5",
        "phone" : "",
        "pwd" : "",
        "pwdLifeTime" : "no-password-expire",
        "pwdSet" : "yes",
        "replTs" : "2013-05-30T11:28:33.835",
        "rn" : "",
        "status" : "created"
    }
}
]
}

```

Because multiple active subscriptions can exist for a query, a notification can contain multiple subscription IDs as in the example shown.



Note Notifications are supported in either JSON or XML format.

Refreshing the Subscription

To continue to receive event notifications, you must periodically refresh each subscription during your API session. To refresh a subscription, send an HTTP GET message to the API method **subscriptionRefresh** with the parameter **id** equal to the **subscriptionId** as in this example:

```
GET https://apic-ip-address/api/subscriptionRefresh.json?id=72057611234574337
```

The API returns an empty response to the refresh message unless the subscription has expired.



Note The timeout period for a subscription is one minute. To prevent lost notifications, you must send a subscription refresh message at least once every 60 seconds.

REST API Examples

Information About the API Examples

In the examples, the JSON and XML structures have been expanded with line feeds, spaces, and indentations for readability.

Example: Using the JSON API to Add a Leaf Port Selector Profile

This example shows how to add a leaf port selector profile.

As shown in the *Cisco APIC Management Information Model Reference*, this hierarchy of classes forms a leaf port selector profile:

- fabric:LePortP — A leaf port profile is represented by a managed object (MO) of this class, which has a distinguished name (DN) format of `uni/fabric/leportp-[name]`, in which `leportp-[name]` is the relative name (RN). The leaf port profile object is a template that can contain a leaf port selector as a child object.
 - fabric:LFPortS — A leaf port selector is represented by an MO of this class, which has a RN format of `lefabports-[name]-typ-[type]`. The leaf port selector object can contain one or more ports or ranges of ports as child objects.
 - fabric:PortBlk — A leaf port or a range of leaf ports is represented by an MO of this class, which has a RN format of `portblk-[name]`.

The API command that creates the new leaf port selector profile MO can also create and configure the child MOs.

This example creates a leaf port selector profile with the name "MyLPSelectorProf." The example profile contains a selector named "MySelectorName" that selects leaf port 1 on leaf switch 1 and leaf ports 3 through 5 on leaf switch 1. To create and configure the new profile, send this HTTP POST message:

```
POST http://apic-ip-address/api/mo/uni/fabric/leportp-MyLPSelectorProf.json
```

```
{
  "fabricLePortP" : {
    "attributes" : {
      "descr" : "Selects leaf ports 1/1 and 1/3-5"
    },
    "children" : [{
      "fabricLFPortS" : {
        "attributes" : {
          "name" : "MySelectorName",
          "type" : "range"
        },
        "children" : [{
          "fabricPortBlk" : {
            "attributes" : {
              "fromCard" : "1",
              "toCard" : "1",
              "fromPort" : "1",
              "toPort" : "1",
              "name" : "block2"
            }
          }, {
            "fabricPortBlk" : {
              "attributes" : {
                "fromCard" : "1",
                "toCard" : "1",
                "fromPort" : "3",
                "toPort" : "5",
                "name" : "block3"
              }
            }
          }
        ]
      }
    ]
  }
}
```

```
}
}
```

A successful operation returns this response body:

```
{
  "imdata" : [{
    "fabricLePortP" : {
      "attributes" : {
        "instanceId" : "0:0",
        "childAction" : "deleteNonPresent",
        "descr" : "Select leaf ports 1/1 and 1/3-5",
        "dn" : "uni/fabric/leportp-MyLPSelectorProf",
        "lcOwn" : "local",
        "name" : "MyLPSelectorProf",
        "replTs" : "never",
        "rn" : "",
        "status" : "created"
      },
      "children" : [{
        "fabricLFPortS" : {
          "attributes" : {
            "instanceId" : "0:0",
            "childAction" : "deleteNonPresent",
            "dn" : "",
            "lcOwn" : "local",
            "name" : "MySelectorName",
            "replTs" : "never",
            "rn" : "leafports-MySelectorName-typ-range",
            "status" : "created",
            "type" : "range"
          },
          "children" : [{
            "fabricPortBlk" : {
              "attributes" : {
                "instanceId" : "0:0",
                "childAction" : "deleteNonPresent",
                "dn" : "",
                "fromCard" : "1",
                "fromPort" : "3",
                "lcOwn" : "local",
                "name" : "block3",
                "replTs" : "never",
                "rn" : "portblk-block3",
                "status" : "created",
                "toCard" : "1",
                "toPort" : "5"
              }
            }
          ],
          "fabricPortBlk" : {
            "attributes" : {
              "instanceId" : "0:0",
              "childAction" : "deleteNonPresent",
              "dn" : "",
              "fromCard" : "1",
              "fromPort" : "1",
              "lcOwn" : "local",
              "name" : "block2",
              "replTs" : "never",
              "rn" : "portblk-block2",
              "status" : "created",
              "toCard" : "1",
```

Example: Using the JSON API to Get Information About a Node

```

        "toPort" : "1"
      }
    }
  ]
}

```

To delete the new profile, send an HTTP POST message with a fabricLePortP attribute of "status": "deleted", as in this example:

```

POST http://apic-ip-address/api/mo/uni/fabric/leportp-MyLPSelectorProf.json

{
  "fabricLePortP" : {
    "attributes" : {
      "status" : "deleted"
    }
  }
}

```

Alternatively, you can send this HTTP DELETE message:

```

DELETE http://apic-ip-address/api/mo/uni/fabric/leportp-MyLPSelectorProf.json

```

Example: Using the JSON API to Get Information About a Node

This example shows how to query the APIC to access a node in the system.

To direct an API operation to a specific node device in the fabric, the resource path consists of `/mo/topology/pod-number/node-number/sys/` followed by the node component. For example, this URI accesses board sensor 3 in chassis slot B of node 1:

```

GET http://apic-ip-address/api/mo/topology/pod-1/node-1/sys/ch/bslot/board/sensor-3.json

```

A successful operation returns a response body similar to this example:

```

{
  "imdata" :
  [
    {
      "eqptSensor" : {
        "attributes" : {
          "instanceId" : "0:0",
          "childAction" : "",
          "dn" : "topology/pod-1/node-1/sys/ch/bslot/board/sensor-3",
          "id" : "3",
          "majorThresh" : "0",
          "mfgTm" : "not-applicable",
          "minorThresh" : "0",
          "model" : "",
          "monPolDn" : ""
        }
      }
    }
  ]
}

```



```

        "rev" : "0",
        "ser" : "",
        "status" : "",
        "type" : "dim",
        "vendor" : "Cisco Systems, Inc."
    }
}
]
}

```

Example: Using the JSON API to Get Running Firmware

This example shows how to query the APIC to determine which firmware images are running.

The detailed information on running firmware is contained in an object of class `firmware:CtrlrRunning`, which is a child class (subtree) of the APIC firmware status container class `firmware:CtrlrFwStatusCont`. Because there can be multiple running firmware instances (one per APIC instance), you can query the container class and filter the response for the subtree of running firmware objects.

This example shows the API query message:

```

GET http://apic-ip-address/api/class/firmware:CtrlrFwStatusCont.json?
  query-target=subtree
  &
  target-subtree-class=firmwareCtrlrRunning

```

A successful operation returns a response body similar to this example:

```

{
  "imdata" : [{
    "firmwareCtrlrRunning" : {
      "attributes" : {
        "instanceId" : "0:0",
        "applId" : "3",
        "childAction" : "",
        "dn" : "CtrlrFwStatusCont/ctrlrRunning-3",
        "lcOwn" : "local",
        "replTs" : "never",
        "rn" : "",
        "status" : "",
        "ts" : "2012-12-31T16:00:00.000",
        "type" : "ifc",
        "version" : "1.1"
      }
    }
  }, {
    "firmwareCtrlrRunning" : {
      "attributes" : {
        "instanceId" : "0:0",
        "applId" : "1",
        "childAction" : "",
        "dn" : "CtrlrFwStatusCont/ctrlrRunning-1",
        "lcOwn" : "local",
        "replTs" : "never",
        "rn" : "",
        "status" : "",
        "ts" : "2012-12-31T16:00:00.000",
        "type" : "ifc",

```

Example: Using the JSON API to Get Top Level System Elements

```

        "version" : "1.1"
      }
    }, {
      "firmwareCtrlrRunning" : {
        "attributes" : {
          "instanceId" : "0:0",
          "applId" : "2",
          "childAction" : "",
          "dn" : "ctrlrfwstatuscont/ctrlrrunning-2",
          "lcOwn" : "local",
          "replTs" : "never",
          "rn" : "",
          "status" : "",
          "ts" : "2012-12-31T16:00:00.000",
          "type" : "ifc",
          "version" : "1.1"
        }
      }
    }
  ]
}

```

This response describes three running instances of APIC firmware version 1.1.

Example: Using the JSON API to Get Top Level System Elements

This example shows how to query the APIC to determine what system devices are present.

General information about the system elements (APICs, spines, and leafs) is contained in an object of class `top:System`.

This example shows the API query message:

```
GET http://apic-ip-address/api/class/topSystem.json
```

A successful operation returns a response body similar to this example:

```

{
  "imdata" :
  [
    {
      "topSystem" : {
        "attributes" : {
          "instanceId" : "0:0",
          "address" : "10.0.0.32",
          "childAction" : "",
          "currentTime" : "2013-06-14T04:13:05.584",
          "currentTimeZone" : "",
          "dn" : "topology/pod-1/node-17/sys",
          "fabricId" : "0",
          "id" : "17",
          "inbMgmtAddr" : "0.0.0.0",
          "lcOwn" : "local",
          "mode" : "unspecified",
          "name" : "leaf0",
          "nodeId" : "0",
          "oobMgmtAddr" : "0.0.0.0",
          "podId" : "1",
          "replTs" : "never",

```

```

        "role" : "leaf",
        "serial" : "FOX-270308",
        "status" : "",
        "systemUpTime" : "00:00:02:03"
    }
}
}, {
    "topSystem" : {
        "attributes" : {
            "instanceId" : "0:0",
            "address" : "10.0.0.1",
            "childAction" : "",
            "currentTime" : "2013-06-14T04:13:29.301",
            "currentTimeZone" : "PDT",
            "dn" : "topology/pod-1/node-1/sys",
            "fabricId" : "0",
            "id" : "1",
            "inbMgmtAddr" : "0.0.0.0",
            "lcOwn" : "local",
            "mode" : "unspecified",
            "name" : "apic0",
            "nodeId" : "0",
            "oobMgmtAddr" : "0.0.0.0",
            "podId" : "0",
            "replTs" : "never",
            "role" : "apic",
            "serial" : "",
            "status" : "",
            "systemUpTime" : "00:00:02:26"
        }
    }
}
]
}

```

This response indicates that the system consists of one APIC (node-1) and one leaf (node-17).

Example: Using the XML API and OwnerTag to Add Audit Log Information to Actions

This example shows how to use the **ownerTag** or **ownerKey** property to add custom audit log information when an object is created or modified.

All configurable objects contain the properties **ownerTag** and **ownerKey**, which are user-configurable string properties. When any configurable object is created or modified by a user action, an audit log record object (**aaa:ModLR**) is automatically created to contain information about the change to be reported in the audit log. The audit log record object includes a list (the **changeSet** property) of the configured object's properties that were changed by the action. In the command to create or modify the configurable object, you can add your own specific tracking information, such as a job ticket number or the name of the person making the change, to the **ownerTag** or **ownerKey** property of the configurable object. This tracking information will then be included in the audit log record along with the details of the change.



Note The **ownerTag** information will appear in the log only when the **ownerTag** contents have been changed. To include the same information in a subsequent configuration change, you can clear the **ownerTag** contents before making the next configuration change. This condition applies also to the **ownerKey** property.

In the following example, a domain reference is added to a tenant configuration. As part of the command, the operator's name is entered as the **ownerKey** and a job number is entered as the **ownerTag**.

```
POST https://apic-ip-address/api/mo/uni/tn-ExampleCorp.xml
<fvTenant name="ExampleCorp" ownerKey="georgewa" ownerTag="chg:00033">
  <aaaDomainRef name="ExampleDomain" ownerKey="georgewa" ownerTag="chg:00033"/>
</fvTenant>
```

In this case, two **aaa:ModLR** records are generated — one for the **fv:Tenant** object and one for the **aaa:DomainRef** object. Unless the **ownerKey** or **ownerTag** properties are unchanged from a previous configuration, their new values will appear in the **changeSet** list of the **aaa:ModLR** records, and this information will appear in the audit log record that reports this configuration change.

Example: XML Get Endpoints (Devices) with IP and MAC Addresses

The **fvCep** class can be used to derive a list of endpoints (devices) attached to the fabric and the associated IP and MAC address and the encapsulation for each object.

Use an XML query, such as the following example, to get a list of endpoints with the IP and MAC address for each one:

Example:

```
GET https://apic-ip-address/api/node/class/fvCep.xml
```

Example: Monitoring Using the REST API

In the examples in this topic, the JSON and XML structures have been expanded with line feeds, spaces, and indentations for readability.

XML Example: Get the Current List of Faults in the Fabric

You can use the **faultInst** class to derive all faults associated with the fabric, tenant, or individual managed objects within the APIC. Send a query with XML such as this example:

```
GET https://apic-ip-address/api/node/class/faultInst.xml?
query-target-filter=and(eq(faultInst.cause,"config-failure"))
```

XML Example: Get the Current List of Faults in the Fabric That Were Caused by a Failed Configuration

You can also use the **fault Inst** class with filters to limit the response to faults that were caused by a failed configuration, with XML such as this example:

```
GET https://apic-ip-address/api/node/class/faultInst.xml?
query-target-filter=and(e(stultification,"config-failure"))
```

XML Example: Get the Properties for a Specific Managed Object, DN

You can use a MO query to obtain the properties of the tenant name, with XML such as the following example:

```
GET https://apic-ip-address/api/node/mo/uni/tn-common.xml?query-target=self
```

Accessing the REST API

Accessing the REST API

By using a script or a browser-based REST client, you can send an API POST or GET message of the form: **https://apic-ip-address /api/ api-message-url**

Use the out-of-band management IP address that you configured during the initial setup.

- Note**
- Only https is enabled by default. By default, http and http-to-https redirection are disabled.
 - You must send an authentication message to initiate an API session. Use the administrator login name and password that you configured during the initial setup.

Invoking the API

You can invoke an API function by sending an HTTP/1.1 or HTTPS POST, GET, or DELETE message to the Application Policy Infrastructure Controller (APIC). The HTML body of the POST message contains a Javascript Object Notation (JSON) or XML data structure that describes an MO or an API method. The HTML body of the response message contains a JSON or XML structure that contains requested data, confirmation of a requested action, or error information.



- Note** The root element of the response structure is imdata. This element is merely a container for the response; it is not a class in the management information model (MIM).

Configuring the HTTP Request Method and Content Type

API commands and queries must use the supported HTTP or HTTPS request methods and header fields, as described in the following sections.



- Note** For security, only HTTPS is enabled as the default mode for API communications. HTTP and HTTP-to-HTTPS redirection can be enabled if desired, but are less secure. For simplicity, this document refers to HTTP in descriptions of protocol components and interactions.

Request Methods

The API supports HTTP POST, GET, and DELETE request methods as follows:

- An API command to create or update an MO, or to execute a method, is sent as an HTTP POST message.

- An API query to read the properties and status of an MO, or to discover objects, is sent as an HTTP GET message.
- An API command to delete an MO is sent as either an HTTP POST or DELETE message. In most cases, you can delete an MO by setting its status to deleted in a POST operation.

Other HTTP methods, such as PUT, are not supported.



Note Although the DELETE method is supported, the HTTP header might show only these:
Access-Control-Allow-Methods: POST, GET, OPTIONS

Content Type

The API supports either JSON or XML data structures in the HTML body of an API request or response. You must specify the content type by terminating the URI pathname with a suffix of either .json or .xml to indicate the format to be used. The HTTP **Content-Type** and **Accept** headers are ignored by the API.

Configuring HTTP and HTTPS Using the GUI

This procedure configures the supported communication protocol for access to the GUI and the REST API.

By default, only HTTPS is enabled. HTTP or HTTP-to-HTTPS redirection, if desired, must be explicitly enabled and configured. HTTP and HTTPS can coexist.

-
- Step 1** On the menu bar, click **Fabric > Fabric Policies**.
 - Step 2** In the **Navigation** pane, expand **Policies > Pod > Management Access**.
 - Step 3** Under **Management Access**, click the default policy.
 - Step 4** In the **Work** pane, in the HTTP or HTTPS areas, enable or disable the protocol by selecting the desired state from the **Admin State** drop-down list.
 - Step 5** In the HTTP area, enable or disable HTTP-to-HTTPS redirection by selecting the desired state from the **Redirect** drop-down list.
 - Step 6** Click **Submit**.
-

Configuring HTTP and HTTPS Throttling Using the CLI

This procedure limits the rate of HTTP and HTTPS requests to the GUI and the REST API.



Note This procedure describes how to configure HTTP and HTTPS AAA login throttling. For information on configuring the NGINX rate limit (global throttling), see the [Cisco ACI Support for NGINX Rate Limit](#) document.

Procedure

	Command or Action	Purpose
Step 1	configure terminal Example: apic1# configure terminal apic1(config)#	Enter configuration mode.
Step 2	comm-policy <i>policy-name</i> Example: apic1(config)# comm-policy default apic1(config-comm-policy)#	Create a new communications policy or edit an existing policy, such as the default policy.
Step 3	http https Example: apic1(config-comm-policy)# https apic1(config-https)#	Select HTTP or HTTPS for throttling configuration.
Step 4	[no] enable-throttle Example: apic1(config-https)# enable-throttle apic1(config-https)#	Enable or disable throttling on the selected protocol. The no prefix disables throttling.
Step 5	throttle <i>1-100</i> Example: apic1(config-https)# throttle 50 apic1(config-https)#	Set the maximum rate of requests per second.

Configuring a Custom Certificate for Cisco ACI HTTPS Access Using the GUI

CAUTION: PERFORM THIS TASK ONLY DURING A MAINTENANCE WINDOW AS THERE IS A POTENTIAL FOR DOWNTIME. The downtime affects access to the Cisco Application Policy Infrastructure Controller (APIC) cluster and switches from external users or systems and not the Cisco APIC to switch connectivity. The NGINX process on the switches will also be impacted, but that will be only for external connectivity and not for the fabric data plane. Access to the Cisco APIC, configuration, management, troubleshooting, and such will be impacted. The NGINX web server running on the Cisco APIC and switches will be restarted during this operation.

Before you begin

Determine from which authority you will obtain the trusted certification so that you can create the appropriate Certificate Authority.

Step 1 On the menu bar, choose **Admin > AAA**.

- Step 2** In the **Navigation** pane, choose **Security**.
- Step 3** In the **Work** pane, choose **Public Key Management > Certificate Authorities > Create Certificate Authority**.
- Step 4** In the **Create Certificate Authority** dialog box, in the **Name** field, enter a name for the certificate authority.
- Step 5** In the **Certificate Chain** field, copy the intermediate and root certificates for the certificate authority that will sign the Certificate Signing Request (CSR) for the Cisco APIC.

The certificate should be in Base64 encoded X.509 (CER) format. The intermediate certificate is placed before the root CA certificate. It should look similar to the following example:

```
-----BEGIN CERTIFICATE-----
<Intermediate Certificate>
-----END CERTIFICATE-----
-----BEGIN CERTIFICATE-----
<Root CA Certificate>
-----END CERTIFICATE-----
```

- Step 6** Click **Submit**.
- Step 7** In the **Navigation** pane, choose **Public Key Management > Key Rings**.
- Step 8** In the **Work** pane, choose **Actions > Create Key Ring**.
- The key ring enables you to manage a private key (imported from external device or internally generated on APIC), a CSR generated by the private key, and the certificate signed via the CSR.
- Step 9** In the **Create Key Ring** dialog box, in the **Name** field, enter a name.
- Step 10** In the **Certificate** field, do not add any content if you will generate a CSR using the Cisco APIC through the key ring. Alternately, add the signed certificate content if you already have one that was signed by the CA from the previous steps by generating a private key and CSR outside of the Cisco APIC.
- Step 11** In the **Modulus** field, click the radio button for the desired key strength.
- Step 12** In the **Certificate Authority** field, from the drop-down list, choose the certificate authority that you created earlier, then click **Submit**.
- Step 13** In the **Private Key** field, do not add any content if you will generate a CSR using the Cisco APIC through the key ring. Alternately, add the private key used to generate the CSR for the signed certificate that you entered in step 10.

Note Do not delete the key ring. Deleting the key ring will automatically delete the associated private key used with CSRs.

If you have not entered the signed certificate and the private key, in the **Work** pane, in the **Key Rings** area, the **Admin State** for the key ring created displays **Started**, waiting for you to generate a CSR. Proceed to step 14.

If you entered both the signed certificate and the private key, in the **Key Rings** area, the **Admin State** for the key ring created displays **Completed**. Proceed to step 23.

- Step 14** In the **Navigation** pane, choose **Public Key Management > Key Rings > key_ring_name**.
- Step 15** In the **Work** pane, choose **Actions > Create Certificate Request**.
- Step 16** In the **Subject** field, enter the common name (CN) of the CSR.

You can enter the fully qualified domain name (FQDN) of the Cisco APICs using a wildcard, but in a modern certificate, we generally recommend that you enter an identifiable name of the certificate and enter the FQDN of all Cisco APICs in the **Alternate Subject Name** field (also known as the *SAN* – Subject Alternative Name) because many modern browsers expect the FQDN in the SAN field.

- Step 17** In the **Alternate Subject Name** field, enter the FQDN of all Cisco APICs, such as "DNS:apic1.example.com,DNS:apic2.example.com,DNS:apic3.example.com" or "DNS:*example.com".

Alternatively, if you want SAN to match an IP address, enter the Cisco APICs' IP addresses with the following format:

```
IP:192.168.2.1
```

You can use DNS names, IPv4 addresses, or a mixture of both in this field. IPv6 addresses are not supported.

Step 18 Fill in the remaining fields as appropriate.

Note Check the online help information available in the **Create Certificate Request** dialog box for a description of the available parameters.

Step 19 Click **Submit**.

Inside the same key ring, the **Associated Certificate Request** area is now displayed with the **Subject**, **Alternate Subject Name** and other fields you entered in the previous steps along with the new field **Request**, which contains the content of the CSR that is tied to this key ring. Copy the content from the **Request** field to submit the content to the same certificate authority that is tied to this key ring for signing.

Step 20 In the **Navigation** pane, choose **Public Key Management > Key Rings > key_ring_name**.

Step 21 In the **Work** pane, in the **Certificate** field, paste the signed certificate that you received from the certificate authority.

Step 22 Click **Submit**.

Note If the CSR was not signed by the Certificate Authority indicated in the key ring, or if the certificate has MS-DOS line endings, an error message is displayed and the certificate is not accepted. Remove the MS-DOS line endings.

The key is verified, and in the **Work** pane, the **Admin State** changes to **Completed** and is now ready for use in the HTTP policy.

Step 23 On the menu bar, choose **Fabric > Fabric Policies**.

Step 24 In the **Navigation** pane, choose **Pod Policies > Policies > Management Access > default**.

Step 25 In the **Work** pane, in the **Admin Key Ring** drop-down list, choose the desired key ring.

Step 26 (Optional) For Certificate based authentication, in the **Client Certificate TP** drop-down list, choose the previously created Local User policy and click **Enabled** for **Client Certificate Authentication state**.

Step 27 Click **Submit**.

All web servers restart. The certificate is activated, and the non-default key ring is associated with HTTPS access.

What to do next

You must remain aware of the expiration date of the certificate and take action before it expires. To preserve the same key pair for the renewed certificate, you must preserve the CSR as it contains the public key that pairs with the private key in the key ring. Before the certificate expires, the same CSR must be resubmitted. Do not delete or create a new key ring as deleting the key ring will delete the private key stored internally on the Cisco APIC.

Authenticating and Maintaining an API Session

Before you can access the API, you must first log in with the name and password of a configured user.

When a login message is accepted, the API returns a data structure that includes a session timeout period in seconds and a token that represents the session. The token is also returned as a cookie in the HTTP response header. To maintain your session, you must send an **aaaRefresh** message as a POST or GET to the API, as

described below, where the message is sent within the session timeout period. The token changes each time that the session is refreshed.



Note The default session timeout period is 300 seconds or 5 minutes.

These API methods enable you to manage session authentication:

- **aaaLogin** —Sent as a POST message, this method logs in a user and opens a session. The message body contains an aaa:User object with the name and password attributes, and the response contains a session token and cookie. If multiple AAA login domains are configured, you must prepend the user's name with **apic: domain **.
- **aaaRefresh** —Sent as a GET message with no message body or as a POST message with the **aaaLogin** message body, this method resets the session timer. The response contains a new session token and cookie.
- **aaaLogout** —Sent as a POST message, this method logs out the user and closes the session. The message body contains an aaa:User object with the name attribute. The response contains an empty data structure.
- **aaaListDomains** —Sent as a GET message, this method returns a list of valid AAA login domains. You can send this message without logging in.

You can call the authentication methods using this syntax, specifying either JSON or XML data structures:

```
{ http | https } :// host [:port] /api/ methodName . { json | xml }
```

This example shows a user login message that uses a JSON data structure:

```
POST https://apic-ip-address/api/aaaLogin.json
```

```
{
  "aaaUser" : {
    "attributes" : {
      "name" : "georgewa",
      "pwd" : "paSSword1"
    }
  }
}
```

This example shows part of the response upon a successful login, including the token and the refresh timeout period:

```
RESPONSE:
{
  "imdata" : [{
    "aaaLogin" : {
      "attributes" : {
        "token" :
          "GkZ15NLRZJl5+jqChouaZ9CYjgE58W/pMccR+LeXmd00obG9NB
          Iwo1VBo7+YCl0iJL9mS6I9qh62BkX+Xddhe0JYrTmSG4JcKZ4t3
          bcP2Mxy3VBmgoJjwZ76ZOuf9V9AD6X1831yoR4bLBzqbSSU1R2N
          IgUotCGWjZt5JX6CJF0=",
        "refreshTimeoutSeconds" : "300",
        "lastName" : "Washington",
        "firstName" : "George"
      }
    }
  ]
}
```

```

        },
        "children" : [{
            ...
        ] [TRUNCATED]
        ...
    }

```

In the preceding example, the **refreshTimeoutSeconds** attribute indicates that the session timeout period is 300 seconds.

This example shows how to request a list of valid login domains:

```
GET https://apic-ip-address/api/aaaListDomains.json
```

RESPONSE:

```

{
  "imdata": [{
    "name": "ExampleRadius"
  },
  {
    "name": "local",
    "guiBanner": "San Jose Fabric"
  }
]]
}

```

In the preceding example, the response data shows two possible login domains, 'ExampleRadius' and 'local.' The following example shows a user login message for the ExampleRadius login domain:

```
POST https://apic-ip-address/api/aaaLogin.json
```

```

{
  "aaaUser" : {
    "attributes" : {
      "name" : "apic:ExampleRadius\georgewa",
      "pwd" : "paSSword1"
    }
  }
}

```

Requiring a Challenge Token for an API Session

For stronger API session security, you can require your session to use a challenge token. When you request this feature during login, the API returns a token string that you must include in every subsequent message to the API. Unlike the normal session token, the challenge token is not stored as a cookie to be automatically provided by your browser. Your API commands and queries must provide the challenge token using one of the following methods:

- The challenge token is sent as a 'challenge' parameter in the URI of your API message.
- The challenge token is part of the HTTP or HTTPS header using 'APIC-challenge'.

To initiate a session that requires a challenge token, include the URI parameter statement `?gui-token-request=yes` in your login message, as shown in this example:

```
POST https://192.0.20.123/api/aaaLogin.json?gui-token-request=yes
```

The response message body contains an attribute of the form "urlToken":"*token*", where *token* is a long string of characters representing the challenge token. All subsequent messages to the API during this session must include the challenge token, as shown in this example where it is sent as a 'challenge' URI parameter:

```
GET https://192.0.20.123/api/class/aaaUser.json?challenge=fa47e44df54562c24fef6601dc...
```

This example shows how the challenge token is sent as an 'APIC-challenge' field in the HTTP header:

```
GET //api/class/aaaUser.json
HTTP/1.1
Host: 192.0.20.123
Connection: keep-alive
Accept: text/html,application/xhtml+xml,application/xml,application/json
APIC-challenge: fa47e44df54562c24fef6601dcff72259299a077336aecfc5b012b036797ab0f
.
.
.
```

Logging In

You can log in to the APIC REST API by sending a valid username and password in a data structure to the **aaaLogin** API method, as described in [Authenticating and Maintaining an API Session, on page 39](#). Following a successful login, you must periodically refresh the session.

The following examples show how to log in as an administrator, refresh the session during configuration, and log out using XML and JSON.



Note At this time, the **aaaLogout** method returns a response but does not end a session. Your session ends after a refresh timeout when you stop sending **aaaRefresh** messages.

Changing Your Own User Credentials

When logged in to APIC, you can change your own user credentials, including your password, SSH key, and X.509 certificate. The following API methods are provided for changing the user credentials of the logged-in user:

- `changeSelfPassword`
- `changeSelfSshKey`
- `changeSelfX509Cert`



Note Using these methods, you can change the credentials only for the account under which you are logged in.

The message body of each method contains the properties of the object to be modified. The properties are shown in the *Cisco APIC Management Information Model Reference*.

Changing Your Password

To change your password, send the `changeSelfPassword` API method, which modifies the `aaa:changePassword` object. The following object properties are required in the message body:

- `userName` — Your login ID.
- `oldPassword` — Your current password.
- `newPassword` — Your new password.

This example, when sent by `User1`, changes the password for `User1`.

```
POST http://192.0.20.123/api/changeSelfPassword.json

{
  "aaaChangePassword" : {
    "attributes" : {
      "userName" : "User1",
      "oldPassword" : "p@$w0rd",
      "newPassword" : "dr0ws$p"
    }
  }
}
```

A successful operation returns an empty `imdata` element, as in this example:

```
{
  "totalCount" : "0",
  "imdata" : []
}
```

Changing Your SSH Key

To change your SSH key, send the `changeSelfSshKey` API method, which modifies the `aaa:changeSshKey` object. The following object properties are required in the message body:

- `userName` — Your login ID.
- `name` — The symbolic name of the key. APIC supports up to 32 SSH keys for a single user.
- `data` — Your new SSH key.

This example, when sent by `User1`, changes the SSH key for `User1`.

```
POST http://192.0.20.123/api/changeSelfSshKey.json

{
  "aaaChangeSshKey" : {
    "attributes" : {
      "userName" : "User1",
      "name" : "A",
      "data" : "ssh-rsa AAAAB3NzaC1yc2EAAAABIWAAAQEaUKxY5E4we6uCR2z== key@example.com"
    }
  }
}
```

A successful operation returns an empty `imdata` element.

Changing Your X.509 Certificate

To change your X.509 certificate, send the `changeSelfX509Cert` API method, which modifies the `aaa:changeX509Cert` object. The following object properties are required in the message body:

- `userName` — Your login ID.
- `name` — The symbolic name of the certificate. APIC supports up to 32 X.509 certificates for a single user.
- `data` — The entire data body of your new X.509 certificate.

This example, when sent by User1, changes the X.509 certificate for User1.

```
POST http://192.0.20.123/api/changeSelfX509Cert.json

{
  "aaaChangeX509Cert" : {
    "attributes" : {
      "userName" : "User1",
      "name" : "A",
      "data" : "-----BEGIN CERTIFICATE-----\nMIIE2TCCA8GgAwIBAgIKamlnsw
[EXAMPLE TRUNCATED]
      1BCIo1blPFft6QKoSJFjB6thJksaE5/k3Npf\n-----END CERTIFICATE-----"
    }
  }
}
```

A successful operation returns an empty `imdata` element.

Deleting an SSH Key or X.509 Certificate

To delete a key or certificate, send the key or certificate change method with the name of the key or certificate to be deleted and with the data attribute blank.

This example, when sent by User1, deletes the SSH key for User1.

```
POST http://192.0.20.123/api/changeSelfSshKey.json

{
  "aaaChangeSshKey" : {
    "attributes" : {
      "userName" : "User1",
      "name" : "A",
      "data" : ""
    }
  }
}
```

A successful operation returns an empty `imdata` element.


REST API Tools

Management Information Model Reference

The Management Information Model (MIM) contains all of the managed objects in the system and their properties. For details, see the *Cisco APIC Management Information Model Reference Guide*.

See the following figure for an example of how an administrator can use the MIM to research an object in the MIT.

Figure 3: MIM Reference



Management Information Model Reference

All Postages

Classes

- [aaa:AuthProvider](#)
- [aaa:ARefP](#)
- [aaa:AuthMethod](#)
- [aaa:AuthRealm](#)
- [aaa:Banner](#)
- [aaa:ChangePassword](#)
- [aaa:ChangeSshKey](#)
- [aaa:ChangeX509Cert](#)
- [aaa:Config](#)
- [aaa:ConsoleAuth](#)
- [aaa:DefaultAuth](#)
- [aaa:Definition](#)
- [aaa:Domain](#)
- [aaa:DomainAuth](#)
- [aaa:DomainRef](#)
- [aaa:DomainRolesTuple](#)
- [aaa:Ep](#)
- [aaa:IfcRefP](#)
- [aaa:LdapEp](#)
- [aaa:LdapProvider](#)
- [aaa:LdapProviderGroup](#)
- [aaa:LoginDomain](#)
- [aaa:Modi_R](#)
- [aaa:PreLoginBanner](#)
- [aaa:ProviderGroup](#)
- [aaa:ProviderRef](#)
- [aaa:PwdProfile](#)
- [aaa:RadiusEp](#)
- [aaa:RadiusProvider](#)
- [aaa:RadiusProviderGroup](#)
- [aaa:Realm](#)
- [aaa:RemoteUser](#)
- [aaa:Role](#)
- [aaa:RoleRefP](#)

Methods

Types

Events

Faults

FSMs

Errors

Syslog Messages

Overview
Diagram
Inheritance
Stats
Events
Faults
FSMs
Properties
Summary
Properties
Detail

Class aaa:Ep (ABSTRACT)

Class ID:705
 Encrypted: false - Exportable: true - Persistent: true
 Write Access: [aaa, admin, none]
 Read Access: [aaa, admin, none]
 Semantic Scope: None
 Semantic Scope Evaluation Rule: Subclasses
 Monitoring Policy Source: Parent
 Monitoring Flags : [IsObservable: false, HasStats: false, HasFaults: false, HasHealth: false]

The base class for a AAA endpoint. This is an abstract class and cannot be instantiated.

Naming Rules

DN FORGOT:

[0] `uni/username/`

Diagram

LEGEND

C ConcreteModelA

- o admin-prop
- o implicit-prop
- o naming-readonly-prop
- o open-prop

explicit relation

A AbstractModelB

named relation

R RelationModel

- o prop1
- o prop2

C C

C UserEp

- o pwdStrengthCheck : aaa Boolean

A Definition

- o name : naming Name

A Ep

- o name : aaa Name
- o timeout : aaa TimeSec

C LdapEp

- o attribute : aaa LdapAttribute
- o basedn : aaa LdapDN
- o filter : aaa LdapFilter
- o timeout : aaa TimeSec

C RadiusEp

C TacacsPlusEp

348553

Cisco APIC REST API Configuration Guide, Release 4.1(x)

46

Viewing an API Interchange in the GUI

When you perform a task in the APIC graphical user interface (GUI), the GUI creates and sends internal API messages to the operating system to execute the task. By using the API Inspector, which is a built-in tool of the APIC, you can view and copy these API messages. A network administrator can replicate these messages in order to automate key operations, or you can use the messages as examples to develop external applications that will use the API. .

Step 1 Log in to the APIC GUI.

Step 2 In the upper right corner of the APIC window, click the "welcome, <name>" message to view the drop-down list.

Step 3 In the drop-down list, choose the **Show API Inspector**.

The **API Inspector** opens in a new browser window.

Step 4 In the **Filters** toolbar of the **API Inspector** window, choose the types of API log messages to display.

The displayed messages are color-coded according to the selected message types. This table shows the available message types:

Name	Description
trace	Displays trace messages.
debug	Displays debug messages. This type includes most API commands and responses.
info	Displays informational messages.
warn	Displays warning messages.
error	Displays error messages.
fatal	Displays fatal messages.
all	Checking this checkbox causes all other checkboxes to become checked. Unchecking any other checkbox causes this checkbox to be unchecked.

Step 5 In the **Search** toolbar, you can search the displayed messages for an exact string or by a regular expression.

This table shows the search controls:

Name	Description
Search	In this text box, enter a string for a direct search or enter a regular expression for a regex search. As you type, the first matched field in the log list is highlighted.
Reset	Click this button to clear the contents of the Search text box.
Regex	Check this checkbox to use the contents of the Search text box as a regular expression for a search.
Match case	Check this checkbox to make the search case sensitive.
Disable	Check this checkbox to disable the search and clear the highlighting of search matches in the log list.
Next	Click this button to cause the log list to scroll to the next matched entry. This button appears only when a search is active.

Name	Description
Previous	Click this button to cause the log list to scroll to the previous matched entry. This button appears only when a search is active.
Filter	Check this checkbox to hide nonmatched lines. This checkbox appears only when a search is active.
Highlight all	Check this checkbox to highlight all matched fields. This checkbox appears only when a search is active.

Step 6 In the **Options** toolbar, you can arrange the displayed messages.

This table shows the available options:

Name	Description
Log	Check this checkbox to enable logging.
Wrap	Check this checkbox to enable wrapping of lines to avoid horizontal scrolling of the log list
Newest at the top	Check this checkbox to display log entries in reverse chronological order.
Scroll to latest	Check this checkbox to scroll immediately to the latest log entry.
Clear	Click this button to clear the log list.
Close	Click this button to close the API Inspector.

Example

This example shows two debug messages in the API Inspector window:

```
13:13:36 DEBUG - method: GET url: http://192.0.20.123/api/class/infraInfra.json
response: {"imdata":[{"infraInfra":{"attributes":{"instanceId":"0:0","childAction":"","dn":"uni/infra","lcOwn":"local","name":"","replTs":"never","status":""}}}]}
```

```
13:13:40 DEBUG - method: GET url: http://192.0.20.123/api/class/l3extDomP.json?
query-target=subtree&subscription=yes
response: {"subscriptionId":"72057598349672459","imdata":[]}
```

Testing the API Using Browser Add-Ons

Using a Browser

To test an API request, you can assemble an HTTP message, send it, and inspect the response using a browser add-on utility. RESTful API clients, which are available as add-ons for most popular browsers, provide a user-friendly interface for interacting with the API. Clients include the following:

- For Firefox/Mozilla—Poster, RESTClient
- For Chrome—Advanced REST client, Postman

Browser add-ons pass the session token as a cookie so that there is no need to include the token in the payload data structure.

Testing the API with cURL

You can send API messages from a console or a command-line script using cURL, which is a tool for transferring files using URL syntax.

To send a POST message, create a file that contains the JSON or XML command body, and then enter the cURL command in this format:

```
curl -X POST --data "@<filename>" <URI>
```

You must specify the name of your descriptor file and the URI of the API operation.



Note Make sure to include the "@" symbol before the descriptor filename.

This example creates a new tenant named ExampleCorp using the JSON data structure in the file "newtenant.json":

```
curl -X POST --data "@newtenant.json" https://apic-ip-address/api/mo/uni/tn-ExampleCorp.json
```

To send a GET message, enter the cURL command in this format:

```
curl -X GET <URI>
```

This example reads information about a tenant in JSON format:

```
curl -X GET https://apic-ip-address/api/mo/uni/tn-ExampleCorp.json
```



Note When testing with cURL, you must log in to the API, store the authentication token, and include the token in subsequent API operations.

Related Topics

[Example: Using the JSON API to Add a User with cURL](#)

Cisco APIC Python SDK

The Python API provides a Python programming interface to the underlying REST API, allowing you to develop your own applications to control the APIC and the network fabric, enabling greater flexibility in infrastructure automation, management, monitoring and programmability.

The Python API supports Python version 2.7.

For more information, see *Cisco APIC Python SDK Documentation, Installing the Cisco APIC Python SDK* and <http://www.python-requests.org>.

Using the Managed Object Browser (Visore)

The Managed Object Browser, or Visore, is a utility built into the APIC that provides a graphical view of the managed objects (MOs) using a browser. The Visore utility uses the APIC REST API query methods to browse MOs active in the Application Centric Infrastructure fabric, allowing you to see the query that was used to obtain the information. The Visore utility cannot be used to perform configuration operations.



Note Only the Firefox, Chrome, and Safari browsers are supported for Visore access.

Visore Browser Page

Filter Area

The filter form is case sensitive. This area supports all simple APIC REST API query operations.

Name	Description
Class or DN field	Object class name or fully distinguished name of a managed object.
Property field	The property of the managed object on which you want to filter the results. If you leave the Property field empty, the search returns all instances of the specific class.
Op drop-down list	Operator for the values of the property on which you want to filter the results. The following are valid operators: <ul style="list-style-type: none"> • == (equal to) • != (not equal to) • < (less than) • > (greater than) • ≤ (less than or equal to) • ≥ (greater than or equal to) • between • wildcard • anybit • allbits
Val1 field	The first value for the property on which you want to filter.
Val2 field	The second value on which you want to filter.

Display XML of Last Query Link

The **Display XML of last query** link displays the full APIC REST API translation of the most recent query run in Visore.

Results Area

You can bookmark any query results page in your browser to view the results again because the query is encoded in the URL.



Note Many of the managed objects are only used internally and are not generally applicable to APIC REST API program development.

Name	Description
Pink background	Separates individual managed object instances and displays the class name of the object below it.
Blue or green background	Indicates the property names of the managed object.
Yellow or beige background	Indicates the value of a property name.
dn property	Absolute address of each managed object in the object model.
dn link	When clicked, displays all managed objects with that dn.
Class name link	When clicked, displays all managed objects of that class.
Left arrow	When clicked, takes you to the parent object of the managed object.
Right arrow	When clicked, takes you to the child objects of the managed object.
Question mark	Links you to the XML API documentation for the managed object.

Accessing Visore

Step 1 Open a supported browser and enter the URL of the APIC followed by `/visore.html`.

Example:

```
https://apic-ip-address/visore.html
```

Step 2 When prompted, log in using the same credentials you would use to log in to the APIC CLI or GUI user interfaces. You can use a read-only account.

Running a Query in Visore

- Step 1** Enter a class or DN name of the MO in the **Class or DN** text box.
- Step 2** (Optional) You can filter the query by entering a property of the MO in the **Property** text box, an operator in the **Op** text box, and one or two values in the **Val1** and **Val2** text boxes.
- Step 3** Click **Run Query**.
- Visore sends a query to the APIC and the requested MO is displayed in a tabular format.
- Step 4** (Optional) Click the **Display URI of last query** link to display the API call that executed the query.
- Step 5** (Optional) Click the **Display last response** link to display the API response data structure from the query.
- Step 6** (Optional) In the **dn** field of the MO description table, click the < and > icons to retrieve the parent and child classes of the displayed MO.
- Clicking > sends a query to the APIC for the children of the MO. Clicking < sends a query for the parent of the MO.
- Step 7** (Optional) In the **dn** field of the MO description table, click the additional icons to display statistics, faults, or health information for the MO.
-



PART II

Part 2: Common APIC Tasks Using the REST API

- [Managing APIC Using the REST API, on page 55](#)
- [Managing Roles, Users, and Signature-Based Transactions, on page 79](#)
- [Common Tenant Tasks, on page 91](#)
- [Managing Layer 2 Networking, on page 95](#)
- [Managing Layer 3 Networking, on page 101](#)
- [Monitoring Using the REST API, on page 111](#)
- [Troubleshooting Using the REST API, on page 121](#)



CHAPTER 3

Managing APIC Using the REST API

- [Adding Management Access, on page 55](#)
- [Managing Configuration Files, on page 65](#)
- [Snapshots and Rollbacks, on page 71](#)
- [Using Configuration Zones, on page 73](#)

Adding Management Access

In-Band and Out-of-Band Management Access

The mgmt tenant provides a convenient means to configure access to fabric management functions. While fabric management functions are accessible through the APIC, they can also be accessed directly through in-band and out-of-band network policies.

Static and Dynamic Management Access

APIC supports both static and dynamic management access. For simple deployments where users manage the IP addresses of a few leaf and spine switches, configuring static in-band and out-of-band management connectivity is simpler. For more complex deployments, where you might have a large number of leaf and spine switches that require managing many IP addresses, static management access is not recommended. For detailed information about static management access, see *Cisco APIC and Static Management Access*.

About Static Management Access

Configuring static in-band and out-of-band management connectivity is simpler than configuring dynamic in-band and out-of-band management connectivity. When configuring in-band static management, you must specify the IP address for each node and make sure to assign unique IP addresses. For simple deployments where users manage the IP addresses of a few leaf and spine switches, it is easy to configure a static management access. For more complex deployments, where you might have a large number of leaf and spine switches that require managing many IP addresses, static management access is not recommended. We recommend that you configure a dynamic management access that automatically avoids the possible duplication of IP addresses.

Configuring In-Band Management Access Using the REST API

IPv4 and IPv6 addresses are supported for in-band management access. IPv6 configurations are supported using static configurations (for both in-band and out-of-band). IPv4 and IPv6 dual in-band and out-of-band configurations are supported only through static configuration. For more information, see the KB article, *Configuring Static Management Access in Cisco APIC*.

SUMMARY STEPS

1. Create a VLAN namespace.
2. Create a physical domain.
3. Create selectors for the in-band management.
4. Configure an in-band bridge domain and endpoint group (EPG).
5. Create an address pool.
6. Create management groups.

DETAILED STEPS

Step 1 Create a VLAN namespace.

Example:

```
POST
https://apic-ip-address/api/mo/uni.xml

<?xml version="1.0" encoding="UTF-8"?>
<!-- api/policymgr/mo/uni.xml -->
<polUni>
  <infraInfra>
    <!-- Static VLAN range -->
    <fvnsVlanInstP name="inband" allocMode="static">
      <fvnsEncapBlk name="encap" from="vlan-10" to="vlan-11"/>
    </fvnsVlanInstP>
  </infraInfra>
</polUni>
```

Step 2 Create a physical domain.

Example:

```
POST
https://apic-ip-address/api/mo/uni.xml

<?xml version="1.0" encoding="UTF-8"?>
<!-- api/policymgr/mo/uni.xml -->
<polUni>
  <physDomP name="inband">
    <infraRsVlanNs tDn="uni/infra/vlanns-inband-static"/>
  </physDomP>
</polUni>
```

Step 3 Create selectors for the in-band management.

Example:

```
POST
https://apic-ip-address/api/mo/uni.xml

<?xml version="1.0" encoding="UTF-8"?>
```

```

<!-- api/policymgr/mo/.xml -->
<polUni>
  <infraInfra>
    <infraNodeP name="vmmNodes">
      <infraLeafS name="leafS" type="range">
        <infraNodeBlk name="single0" from_="101" to_="101"/>
      </infraLeafS>
      <infraRsAccPortP tDn="uni/infra/accportprof-vmmPorts"/>
    </infraNodeP>

    <!-- Assumption is that VMM host is reachable via eth1/40. -->
    <infraAccPortP name="vmmPorts">
      <infraHPorts name="portS" type="range">
        <infraPortBlk name="block1"
          fromCard="1" toCard="1"
          fromPort="40" toPort="40"/>
        <infraRsAccBaseGrp tDn="uni/infra/funcprof/accportgrp-inband" />
      </infraHPorts>
    </infraAccPortP>

    <infraNodeP name="apicConnectedNodes">
      <infraLeafS name="leafS" type="range">
        <infraNodeBlk name="single0" from_="101" to_="102"/>
      </infraLeafS>
      <infraRsAccPortP tDn="uni/infra/accportprof-apicConnectedPorts"/>
    </infraNodeP>

    <!-- Assumption is that APIC is connected to eth1/1. -->
    <infraAccPortP name="apicConnectedPorts">
      <infraHPorts name="portS" type="range">
        <infraPortBlk name="block1"
          fromCard="1" toCard="1"
          fromPort="1" toPort="3"/>
        <infraRsAccBaseGrp tDn="uni/infra/funcprof/accportgrp-inband" />
      </infraHPorts>
    </infraAccPortP>

    <infraFuncP>
      <infraAccPortGrp name="inband">
        <infraRsAttEntP tDn="uni/infra/attentp-inband"/>
      </infraAccPortGrp>
    </infraFuncP>

    <infraAttEntityP name="inband">
      <infraRsDomP tDn="uni/phys-inband"/>
    </infraAttEntityP>
  </infraInfra>
</polUni>

```

Step 4 Configure an in-band bridge domain and endpoint group (EPG).

Example:

```

POST https://apic-ip-address/api/mo/uni.xml

<?xml version="1.0" encoding="UTF-8"?>
<!-- api/policymgr/mo/.xml -->
<polUni>
  <fvTenant name="mgmt">
    <!-- Configure the in-band management gateway address on the
      in-band BD. -->
    <fvBD name="inb">
      <fvSubnet ip="10.13.1.254/24"/>
    </fvBD>
  </fvTenant>
</polUni>

```

```

<mgmtMgmtP name="default">
  <!-- Configure the encap on which APICs will communicate on the
in-band network. -->
  <mgmtInB name="default" encap="vlan-10">
    <fvRsProv tnVzBrCPName="default"/>
  </mgmtInB>
</mgmtMgmtP>
</fvTenant>
</polUni>

```

Step 5 Create an address pool.

Example:

```

POST
https://apic-ip-address/api/mo/uni.xml

<?xml version="1.0" encoding="UTF-8"?>
<!-- api/policymgr/mo/.xml -->
<polUni>
  <fvTenant name="mgmt">
    <!-- Addresses for APIC in-band management network -->
    <fvnsAddrInst name="apicInb" addr="10.13.1.254/24">
      <fvnsUcastAddrBlk from="10.13.1.1" to="10.13.1.10"/>
    </fvnsAddrInst>

    <!-- Addresses for switch in-band management network -->
    <fvnsAddrInst name="switchInb" addr="10.13.1.254/24">
      <fvnsUcastAddrBlk from="10.13.1.101" to="10.13.1.120"/>
    </fvnsAddrInst>
  </fvTenant>
</polUni>

```

Note Dynamic address pools for IPv6 is not supported.

Step 6 Create management groups.

Example:

```

POST
https://apic-ip-address/api/mo/uni.xml

<?xml version="1.0" encoding="UTF-8"?>
<!-- api/policymgr/mo/.xml -->
<polUni>
  <infraInfra>
    <!-- Management node group for APICs -->
    <mgmtNodeGrp name="apic">
      <infraNodeBlk name="all" from_"1" to_"3"/>
      <mgmtRsGrp tDn="uni/infra/funcprof/grp-apic"/>
    </mgmtNodeGrp>

    <!-- Management node group for switches-->
    <mgmtNodeGrp name="switch">
      <infraNodeBlk name="all" from_"101" to_"104"/>
      <mgmtRsGrp tDn="uni/infra/funcprof/grp-switch"/>
    </mgmtNodeGrp>

    <!-- Functional profile -->
    <infraFuncP>
      <!-- Management group for APICs -->
      <mgmtGrp name="apic">
        <!-- In-band management zone -->
        <mgmtInBZone name="default">

```

```

    <mgmtRsInbEpg tDn="uni/tn-mgmt/mgmtp-default/inb-default"/>
    <mgmtRsAddrInst tDn="uni/tn-mgmt/addrinst-apicInb"/>
  </mgmtInBZone>
</mgmtGrp>

<!-- Management group for switches -->
<mgmtGrp name="switch">
  <!-- In-band management zone -->
  <mgmtInBZone name="default">
    <mgmtRsInbEpg tDn="uni/tn-mgmt/mgmtp-default/inb-default"/>
    <mgmtRsAddrInst tDn="uni/tn-mgmt/addrinst-switchInb"/>
  </mgmtInBZone>
</mgmtGrp>
</infraFuncP>
</infraInfra>
</polUni>

```

Note Dynamic address pools for IPv6 is not supported.

Configuring Static In-Band Management Access Using the REST API

Step 1 Create a VLAN namespace.

Example:

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- api/policymgr/mo/uni.xml -->
<polUni>
  <infraInfra>
    <!-- Static VLAN range -->
    <fvnsVlanInstP name="inband" allocMode="static">
      <fvnsEncapBlk name="encap" from="vlan-10" to="vlan-11"/>
    </fvnsVlanInstP>
  </infraInfra>
</polUni>

```

Step 2 Create a physical domain.

Example:

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- api/policymgr/mo/uni.xml -->
<polUni>
  <physDomP name="inband">
    <infraRsVlanNs tDn="uni/infra/vlanns-inband-static"/>
  </physDomP>
</polUni>

```

Step 3 Create selectors for the in-band management.

Example:

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- api/policymgr/mo/.xml -->
<polUni>
  <infraInfra>
    <infraNodeP name="vmmNodes">
      <infraLeafS name="leafs" type="range">
        <infraNodeBlk name="single0" from_"101" to_"101"/>
      </infraLeafS>
    </infraNodeP>
  </infraInfra>
</polUni>

```

```

    </infraLeafS>
    <infraRsAccPortP tDn="uni/infra/accportprof-vmmPorts"/>
</infraNodeP>

<!-- Assumption is that VMM host is reachable via eth1/40. -->
<infraAccPortP name="vmmPorts">
  <infraHPortS name="portS" type="range">
    <infraPortBlk name="block1"
      fromCard="1" toCard="1"
      fromPort="40" toPort="40"/>
    <infraRsAccBaseGrp tDn="uni/infra/funcprof/accportgrp-inband" />
  </infraHPortS>
</infraAccPortP>

<infraNodeP name="apicConnectedNodes">
  <infraLeafS name="leafS" type="range">
    <infraNodeBlk name="single0" from_="101" to_="102"/>
  </infraLeafS>
  <infraRsAccPortP tDn="uni/infra/accportprof-apicConnectedPorts"/>
</infraNodeP>

<!-- Assumption is that APIC is connected to eth1/1. -->
<infraAccPortP name="apicConnectedPorts">
  <infraHPortS name="portS" type="range">
    <infraPortBlk name="block1"
      fromCard="1" toCard="1"
      fromPort="1" toPort="3"/>
    <infraRsAccBaseGrp tDn="uni/infra/funcprof/accportgrp-inband" />
  </infraHPortS>
</infraAccPortP>

<infraFuncP>
  <infraAccPortGrp name="inband">
    <infraRsAttEntP tDn="uni/infra/attentp-inband"/>
  </infraAccPortGrp>
</infraFuncP>

<infraAttEntityP name="inband">
  <infraRsDomP tDn="uni/phys-inband"/>
</infraAttEntityP>
</infraInfra>
</polUni>

```

Step 4 Configure an in-band bridge domain and endpoint group (EPG).

Example:

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- api/policymgr/mo/.xml -->
<polUni>
  <fvTenant name="mgmt">
    <!-- Configure the in-band management gateway address on the
    in-band BD. -->
    <fvBD name="inb">
      <fvSubnet ip="<subnet_ip_address"/>/>
    </fvBD>

    <mgmtMgmtP name="default">
      <!-- Configure the encap on which APICs will communicate on the
      in-band network. -->
      <mgmtInB name="default" encap="vlan-10">
        <fvRsProv tnVzBrCPName="default"/>
      </mgmtInB>
    </mgmtMgmtP>

```

```

    </fvTenant>
  </polUni>

```

Step 5 Create static in-band management IP addresses and assign them to node IDs.

Example:

```

<polUni>
  <fvTenant name="mgmt">
    <mgmtMgmtP name="default">
      <mgmtInB name="default">
        <mgmtRsInBStNode tDn="topology/pod-1/node-101"
          addr="<ip_address_1>"
          gw="<gw_address>"
          v6Addr = "<ip6_address_1>"
          v6Gw = "<ip6_gw_address>"/>
        <mgmtRsInBStNode tDn="topology/pod-1/node-102"
          addr="<ip_address_2>"
          gw="<gw_address>"
          v6Addr = "<ip6_address_2>"
          v6Gw = "<ip6_gw_address>"/>
        <mgmtRsInBStNode tDn="topology/pod-1/node-103"
          addr="<ip_address_3>"
          gw="<gw_address>"
          v6Addr = "<ip6_address_3>"
          v6Gw = "<ip6_gw_address>"/>
        <mgmtRsInBStNode tDn="topology/pod-1/node-104"
          addr="<ip_address_4>"
          gw="<gw_address>"
          v6Addr = "<ip6_address_4>"
          v6Gw = "<ip6_gw_address>"/>
        <mgmtRsInBStNode tDn="topology/pod-1/node-105"
          addr="<ip_address_5>"
          gw="<gw_address>"
          v6Addr = "<ip6_address_5>"
          v6Gw = "<ip6_gw_address>"/>
      </mgmtInB>
    </mgmtMgmtP>
  </fvTenant>
</polUni>

```

Configuring Out-of-Band Management Access Using the REST API

IPv4 and IPv6 addresses are supported for out-of-band management access.

Before you begin

The APIC out-of-band management connection link must be 1 Gbps.

SUMMARY STEPS

1. Create an out-of-band contract.
2. Associate the out-of-band contract with an out-of-band EPG.
3. Associate the out-of-band contract with an external management EPG.
4. Create a management address pool.

5. Create node management groups.

DETAILED STEPS

Step 1 Create an out-of-band contract.

Example:

POST <https://apic-ip-address/api/mo/uni.xml>

```
<polUni>
  <fvTenant name="mgmt">
    <!-- Contract -->
    <vzOOBBrCP name="oob-default">
      <vzSubj name="oob-default">
        <vzRsSubjFiltAtt tnVzFilterName="default" />
      </vzSubj>
    </vzOOBBrCP>
  </fvTenant>
</polUni>
```

Step 2 Associate the out-of-band contract with an out-of-band EPG.

Example:

POST <https://apic-ip-address/api/mo/uni.xml>

```
<polUni>
  <fvTenant name="mgmt">
    <mgmtMgmtP name="default">
      <mgmtOoB name="default">
        <mgmtRsOoBProv tnVzOOBBrCPName="oob-default" />
      </mgmtOoB>
    </mgmtMgmtP>
  </fvTenant>
</polUni>
```

Step 3 Associate the out-of-band contract with an external management EPG.

Example:

POST <https://apic-ip-address/api/mo/uni.xml>

```
<polUni>
  <fvTenant name="mgmt">
    <mgmtExtMgmtEntity name="default">
      <mgmtInstP name="oob-mgmt-ext">
        <mgmtRsOoBCons tnVzOOBBrCPName="oob-default" />
        <!-- SUBNET from where switches are managed -->
        <mgmtSubnet ip="10.0.0.0/8" />
      </mgmtInstP>
    </mgmtExtMgmtEntity>
  </fvTenant>
</polUni>
```

Step 4 Create a management address pool.

Example:

POST <https://apic-ip-address/api/mo/uni.xml>


```

<polUni>
  <fvTenant name="mgmt">
    <fvnsAddrInst name="switchOoboobaddr" addr="172.23.48.1/21">
      <fvnsUcastAddrBlk from="172.23.49.240" to="172.23.49.244"/>
    </fvnsAddrInst>
  </fvTenant>
</polUni>

```

Step 5 Create node management groups.**Example:**

POST <https://apic-ip-address/api/mo/uni.xml>

```

<polUni>
  <infraInfra>
    <infraFuncP>
      <mgmtGrp name="switchOob">
        <mgmtOoBZone name="default">
          <mgmtRsAddrInst tDn="uni/tn-mgmt/addrinst-switchOoboobaddr" />
          <mgmtRsOobEpg tDn="uni/tn-mgmt/mgmtp-default/oob-default" />
        </mgmtOoBZone>
      </mgmtGrp>
    </infraFuncP>
    <mgmtNodeGrp name="switchOob">
      <mgmtRsGrp tDn="uni/infra/funcprof/grp-switchOob" />
      <infraNodeBlk name="default" from_"101" to_"103" />
    </mgmtNodeGrp>
  </infraInfra>
</polUni>

```

Note You can configure the APIC server to use out-of-band management connectivity as the default connectivity mode.

```

POST https://apic-ip-address/api/node/mo/.xml
<polUni>
<fabricInst>
  <mgmtConnectivityPrefs interfacePref="ooband"/>
</fabricInst>
</polUni>

```

Configuring Static Out-of-Band Management Access Using the REST API

Before you begin

The APIC out-of-band management connection link must be 1 Gbps.

Step 1 Create an out-of-band contract.**Example:**

```

<polUni>
  <fvTenant name="mgmt">
    <!-- Contract -->
    <vzOOBBrCP name="oob-default">
      <vzSubj name="oob-default">
        <vzRsSubjFiltAtt tnVzFilterName="default" />
      </vzSubj>
    </vzOOBBrCP>
  </fvTenant>
</polUni>

```

```

    </vzOOBBrCP>
  </fvTenant>
</polUni>

```

Step 2 Associate the out-of-band contract with an out-of-band EPG.

Example:

```

<polUni>
  <fvTenant name="mgmt">
    <mgmtMgmtP name="default">
      <mgmtOoB name="default">
        <mgmtRsOoBProv tnVzOOBBrCPName="oob-default" />
      </mgmtOoB>
    </mgmtMgmtP>
  </fvTenant>
</polUni>

```

Step 3 Associate the out-of-band contract with an external management EPG.

Example:

```

<polUni>
  <fvTenant name="mgmt">
    <mgmtExtMgmtEntity name="default">
      <mgmtInstP name="oob-mgmt-ext">
        <mgmtRsOoBCons tnVzOOBBrCPName="oob-default" />
        <!-- SUBNET from where switches are managed -->
        <mgmtSubnet ip="<mgmt_subnet_ip_address>" />
      </mgmtInstP>
    </mgmtExtMgmtEntity>
  </fvTenant>
</polUni>

```

Step 4 Create static out-of-band management IP addresses and assign them to node IDs.

CHECK IP Addresses

Example:

```

<polUni>
  <fvTenant name="mgmt">
    <mgmtMgmtP name="default">
      <mgmtOoB name="default">
        <mgmtRsOoBStNode tDn="topology/pod-1/node-101"
          addr="<ip_address_1>"
          gw="<gw_address>" />
        <mgmtRsOoBStNode tDn="topology/pod-1/node-102"
          addr="<ip_address_2>"
          gw="<gw_address>" />
        <mgmtRsOoBStNode tDn="topology/pod-1/node-103"
          addr="<ip_address_3>"
          gw="<gw_address>" />
      </mgmtOoB>
    </mgmtMgmtP>
  </fvTenant>
</polUni>

```

Managing Configuration Files

Overview

This topic provides information on:

- How to use configuration Import and Export to recover configuration states to the last known good state using the Cisco APIC
- How to encrypt secure properties of Cisco APIC configuration files

You can do both scheduled and on-demand backups of user configuration. Recovering configuration states (also known as "roll-back") allows you to go back to a known state that was good before. The option for that is called an Atomic Replace. The configuration import policy (configImportP) supports atomic + replace (importMode=atomic, importType=replace). When set to these values, the imported configuration overwrites the existing configuration, and any existing configuration that is not present in the imported file is deleted. As long as you do periodic configuration backups and exports, or explicitly trigger export with a known good configuration, then you can later restore back to this configuration using the following procedures for the CLI, REST API, and GUI.

For more detailed conceptual information about recovering configuration states using the Cisco APIC, please refer to the *Cisco Application Centric Infrastructure Fundamentals Guide*.

The following section provides conceptual information about encrypting secure properties of configuration files:

Backing Up, Restoring, and Rolling Back Configuration Files Workflow

This section describes the workflow of the features for backing up, restoring, and rolling back configuration files. All of the features described in this document follow the same workflow pattern. Once the corresponding policy is configured, **adminSt** must be set to **triggered** in order to trigger the job.

Once triggered, an object of type **configJob** (representing that run) is created under a container object of type **configJobCont**. (The naming property value is set to the policy DN.) The container's **lastJobName** field can be used to determine the last job that was triggered for that policy.



Note Up to five **configJob** objects are kept under a single job container at a time, with each new job triggered. The oldest job is removed to ensure this.

The **configJob** object contains the following information:

- Execution time
- Name of the file being processed/generated
- Status, as follows:
 - Pending
 - Running

- Failed
 - Fail-no-data
 - Success
 - Success-with-warnings
- Details string (failure messages and warnings)
 - Progress percentage = $100 * \text{lastStepIndex} / \text{totalStepCount}$
 - Field `lastStepDescr` indicating what was being done last

About Configuration Export to Controllers

Configuration export extracts user-configurable managed object (MO) trees from all 32 shards in the cluster, writes them into separate files, then compresses them into a tar gzip file. The configuration export then uploads the tar gzip file to a preconfigured remote location (configured through `configRsRemotePath` pointing to a `fileRemotePath` object) or stores it as a **snapshot** on the controller(s).



Note See the Snapshots section for more details.

The `configExportP` policy is configured as follows:

- **name**—Policy name.
- **format**—Format in which the data is stored inside the exported archive (xml or json).
- **targetDn**—The domain name (DN) of the specific object you want to export. (Empty means everything.)
- **snapshot**—When true, the file is stored on the controller; no remote location configuration is needed.
- **includeSecureFields**—Set to true by default, this indicates whether the encrypted fields (passwords, etc.) should be included in the export archive.



Note The `configSnapshot` object is created holding the information about this snapshot. (See the Snapshots section.)

Scheduling Exports

An export policy can be linked with a scheduler, which triggers the export automatically based on a preconfigured schedule. This is done through the `configRsExportScheduler` relation from the policy to a `trigSchedP` object. (See the Sample Configuration section.)



Note A scheduler is optional. A policy can be triggered at any time by setting the `adminSt` to **triggered**.

About Configuration Import to Controller

Configuration import downloads, extracts, parses, analyzes, and applies the specified, previously exported archive one shard at a time in the following order: infra, fabric, tn-common, then everything else. The fileRemotePath configuration is performed the same way as for export (through configRsRemotePath). Importing snapshots is also supported.

The **configImportP** policy is configured as follows:

- **name**—Policy name
- **fileName**—Name of the archive file (not the path file) to be imported
- **importMode**
 - Best-effort mode: Each MO is applied individually, and errors only cause the invalid MOs to be skipped.



Note If the object is not present on the controller, none of the children of the object get configured. Best-effort mode attempts to configure the children of the object.

- Atomic mode: configuration is applied by whole shards. A single error causes the whole shard to be rolled back to its original state.
- **importType**
 - Replace—Current system configuration is replaced with the contents or the archive being imported. (Only atomic mode is supported.)
 - Merge—Nothing is deleted, and archive content is applied on top the existing system configuration.
- **snapshot**—When true, the file is taken from the controller and no remote location configuration is needed.
- **failOnDecryptErrors**—(true by default) The file fails to import if the archive was encrypted with a different key than the one that is currently set up in the system.

Troubleshooting

The following scenarios may need troubleshooting:

- If the generated archive could not be downloaded from the remote location, refer to the Connectivity Issues section.
- If the import succeeded with warnings, check the details.
- If a file could not be parsed, refer to the following scenarios:
 - If the file is not a valid XML or JSON file, check whether the files from the exported archive were manually modified.
 - If an object property has an unknown property or property value, it may be because:
 - The property was removed or an unknown property value was manually entered.
 - The model type range was modified (non-backward compatible model change).

- The naming property list was modified.
- If an MO could not be configured, note the following:
 - Best-effort mode logs the error and skips the MO.
 - Atomic mode logs the error and skips the shard.

Configuration File Encryption

As of release 1.1(2), the secure properties of APIC configuration files can be encrypted by enabling AES-256 encryption. AES encryption is a global configuration option; all secure properties conform to the AES configuration setting. It is not possible to export a subset of the ACI fabric configuration such as a tenant configuration with AES encryption while not encrypting the remainder of the fabric configuration. See the *Cisco Application Centric Infrastructure Fundamentals*, "Secure Properties" chapter for the list of secure properties.

The APIC uses a 16 to 32 character passphrase to generate the AES-256 keys. The APIC GUI displays a hash of the AES passphrase. This hash can be used to see if the same passphrases was used on two ACI fabrics. This hash can be copied to a client computer where it can be compared to the passphrase hash of another ACI fabric to see if they were generated with the same passphrase. The hash cannot be used to reconstruct the original passphrase or the AES-256 keys.

Observe the following guidelines when working with encrypted configuration files:

- Backward compatibility is supported for importing old ACI configurations into ACI fabrics that use the AES encryption configuration option.



Note Reverse compatibility is not supported; configurations exported from ACI fabrics that have enabled AES encryption cannot be imported into older versions of the APIC software.

- Always enable AES encryption when performing fabric backup configuration exports. Doing so will assure that all the secure properties of the configuration will be successfully imported when restoring the fabric.



Note If a fabric backup configuration is exported without AES encryption enabled, none of the secure properties will be included in the export. Since such an unencrypted backup would not include any of the secure properties, it is possible that importing such a file to restore a system could result in the administrator along with all users of the fabric being locked out of the system.

- The AES passphrase that generates the encryption keys cannot be recovered or read by an ACI administrator or any other user. The AES passphrase is not stored. The APIC uses the AES passphrase to generate the AES keys, then discards the passphrase. The AES keys are not exported. The AES keys cannot be recovered since they are not exported and cannot be retrieved via the REST API.

- The same AES-256 passphrase always generates the same AES-256 keys. Configuration export files can be imported into other ACI fabrics that use the same AES passphrase.
- For troubleshooting purposes, export a configuration file that does not contain the encrypted data of the secure properties. Temporarily turning off encryption before performing the configuration export removes the values of all secure properties from the exported configuration. To import such a configuration file that has all secure properties removed, use the import merge mode; do not use the import replace mode. Using the import merge mode will preserve the existing secure properties in the ACI fabric.
- By default, the APIC rejects configuration imports of files that contain fields that cannot be decrypted. Use caution when turning off this setting. Performing a configuration import inappropriately when this default setting is turned off could result in all the passwords of the ACI fabric to be removed upon the import of a configuration file that does not match the AES encryption settings of the fabric.



Note Failure to observe this guideline could result in all users, including fabric administrations, being locked out of the system.

About the fileRemotePath Object

The fileRemotePath object holds the following remote location-path parameters:

- Hostname or IP
- Port
- Protocol: FTP, SCP, and others
- Remote directory (not file path)
- Username
- Password



Note The password must be resubmitted every time changes are made.

Sample Configuration

The following is a sample configuration:

Under **fabricInst** (uni/fabric), enter:

```
<fileRemotePath name="path-name" host="host name or ip" protocol="scp"
remotePath="path/to/some/folder" userName="user-name" userpasswd="password" />
```

Configuring a Remote Location Using the REST API

This procedure explains how to create a remote location using the REST API.

```
<fileRemotePath name="local" host="host or ip" protocol="ftp|scp|sftp" remotePath="path to
folder" userName="uname" userPasswd="pwd" />
```

Configuring Configuration File Export to Controller Using the REST API

Before you begin

Create a remote path and scheduling policy.



Note When providing a remote location, if you set the snapshot to `True`, the backup ignores the remote path and stores the file on the controller.

SUMMARY STEPS

1. Create a configuration export policy by sending a POST request with XML such as the following example.

DETAILED STEPS

Create a configuration export policy by sending a POST request with XML such as the following example.

Example:

```
<configExportP name="policy-name" format="xml" targetDn="/some/dn or empty which means everything"
snapshot="false" adminSt="triggered">
  <configRsRemotePath tnFileRemotePathName="some remote path name" />
  <configRsExportScheduler tnTrigSchedPName="some scheduler name" />
</configExportP>
```

Configuring a Configuration File Import Policy Using the REST API

SUMMARY STEPS

1. Configure a configuration file import policy, send a post with XML such as the following example:

DETAILED STEPS

Configure a configuration file import policy, send a post with XML such as the following example:

Example:

```
<configImportP name="policy-name" fileName="someexportfile.tgz" importMode="atomic"
importType="replace" snapshot="false" adminSt="triggered">
  <configRsRemotePath tnFileRemotePathName="some remote path name" />
</configImportP>
```


Encrypting Configuration Files Using the REST API

SUMMARY STEPS

1. To encrypt a configuration file using the REST API, send a post with XML such as the following example:

DETAILED STEPS

To encrypt a configuration file using the REST API, send a post with XML such as the following example:

Example:

```
https://apic-ip-address/api/mo/uni/fabric.xml
<pkiExportEncryptionKey passphrase="abcdefghijklmnopqrstuvwxy" strongEncryptionEnabled="true"/>
```

Snapshots and Rollbacks

Snapshots

Snapshots are configuration backup archives, stored (and replicated) in a controller managed folder. To create one, an export can be performed with the **snapshot** property set to true. In this case, no remote path configuration is needed. An object of **configSnapshot** type is created to expose the snapshot to the user.

You can create recurring snapshots, which are saved to **Admin > Import/Export > Export Policies > Configuration > defaultAuto**.

configSnapshot objects provide the following:

- file name
- file size
- creation date
- root DN indicating what the snapshot is of (fabric, infra, specific tenant, and so on)
- ability to remove a snapshot (by setting the retire field to true)

To import a snapshot, first create an import policy. Navigate to **Admin > Import/Export** and click **Import Policies**. Right click and choose **Create Configuration Import Policy** to set the import policy attributes.

About Rollbacks

The **configRollbackP** policy is used to undo the changes made between two snapshots. Managed Objects (MOs) are processed as follows:

- Deleted MOs are recreated.
- Created MOs are deleted.

- Modified MOs are reverted.



Note The rollback feature operates only on snapshots. Remote archives are not supported. If you want to use the data in a remote archive, use the snapshot manager to create a snapshot from the data for the rollback. The policy does not require a remote path configuration.

Rollback Workflow

The policy `snapshotOneDn` and `snapshotTwoDn` fields must be set and the first snapshot (S1) must precede snapshot two (S2). Once triggered, snapshots are extracted and analyzed, and the difference between them is calculated and applied.

MOs are located that are:

- Present in S1 but not present in S2—These MOs are deleted and rollback re-creates them.
- Not present in S1 but not present in S2—These MOs are created after S1 and rollback deletes them if:
 - These MOs are not modified after S2 is taken.
 - None of the MO descendants are created or modified after S2 is taken.
- Present in both S1 and S2, but with different property values—These MO properties are reverted to S1, unless the property was modified to a different value after S2 is taken. In this case, it is left as is.

The rollback feature also generates a diff file that contains the configuration generated as a result of these calculations. Applying this configuration is the last step of the rollback process. The content of this file can be retrieved through a special REST API called `readdiff`:

```
apichost/mqapi2/snapshots.readdiff.xml?jobdn=SNAPSHOT_JOB_DN.
```

Rollback (which is difficult to predict) also has a preview mode (set `preview` to `true`), which prevents rollback from making any actual changes. It calculates and generates the diff file, allowing you to preview what exactly is going to happen once the rollback is actually performed.

Diff Tool

Another special REST API is available, which provides diff functionality between two snapshots:

```
apichost/mqapi2/snapshots.diff.xml?s1dn=SNAPSHOT_ONE_DN&s2dn=SNAPSHOT_TWO_DN.
```

Uploading and Downloading Snapshots Using the REST API

The `configSnapshotManagerP` policy allows you to create snapshots from remotely stored export archives. You can attach a remote path to the policy, provide the file name (same as with `configImportP`), set the mode to download, and trigger. The manager downloads the file, analyzes it to make sure that the archive is valid, stores it on the controller, and creates the corresponding `configSnapshot` object. The snapshot manager also allow you to upload a snapshot archive to a remote location. In this case, the mode must be set to upload.

Before you begin

Set up remotely stored archives.

SUMMARY STEPS

1. To download or upload a snapshot policy, send a POST request with XML such as the following:

DETAILED STEPS

To download or upload a snapshot policy, send a POST request with XML such as the following:

Example:

```
<configSnapshotManagerP name="policy-name" fileName="someexportfile.tgz"
  mode="upload|download" adminSt="triggered">
<configRsRemotePath tnFileRemotePathName="some remote path name" />
</configSnapshotManagerP>
```

Configuring and Executing a Configuration Rollback Using the REST API

Before you begin

Create a rollback policy and a snapshot.

SUMMARY STEPS

1. To configure and execute a rollback, send a POST request with XML such as the following:

DETAILED STEPS

To configure and execute a rollback, send a POST request with XML such as the following:

Example:

```
<configRollbackP name="policy-name" snapshotOneDn="dn/of/snapshot/one"
  snapshotOneDn="dn/of/snapshot/two" preview="false" adminSt="triggered" />
```

Using Configuration Zones

Configuration Zones

Configuration zones divide the ACI fabric into different zones that can be updated with configuration changes at different times. This limits the risk of deploying a faulty fabric-wide configuration that might disrupt traffic or even bring the fabric down. An administrator can deploy a configuration to a non-critical zone, and then deploy it to critical zones when satisfied that it is suitable.

The following policies specify configuration zone actions:

- `infrazone:ZoneP` is automatically created upon system upgrade. It cannot be deleted or modified.

- `infracone:Zone` contains one or more pod groups (`PodGrp`) or one or more node groups (`NodeGrp`).



Note You can only choose `PodGrp` or `NodeGrp`; both cannot be chosen.

A node can be part of only one zone (`infracone:Zone`). `NodeGrp` has two properties: name, and deployment mode. The deployment mode property can be:

- `enabled` - Pending updates are sent immediately.
- `disabled` - New updates are postponed.



Note

- Do not upgrade, downgrade, commission, or decommission nodes in a disabled configuration zone.
- Do not do a clean reload or an uplink/downlink port conversion reload of nodes in a disabled configuration zone.

- `triggered` - pending updates are sent immediately, and the deployment mode is automatically reset to the value it had before the change to `triggered`.

When a policy on a given set of nodes is created, modified, or deleted, updates are sent to each node where the policy is deployed. Based on policy class and `infracone` configuration the following happens:

- For policies that do not follow `infracone` configuration, the APIC sends updates immediately to all the fabric nodes.
- For policies that follow `infracone` configuration, the update proceeds according to the `infracone` configuration:
 - If a node is part of an `infracone:Zone`, the update is sent immediately if the deployment mode of the zone is set to `enabled`; otherwise the update is postponed.
 - If a node is not part of an `infracone:Zone`, the update is done immediately, which is the ACI fabric default behavior.

Configuration Zone Supported Policies

The following policies are supported for configuration zones:

```
analytics:CfgSrv
bgp:InstPol
callhome:Group
callhome:InvP
callhome:QueryGroup
cdp:IfPol
cdp:InstPol
comm:Pol
comp:DomP
coop:Pol
datetime:Pol
dbgexp:CoreP
dbgexp:TechSupP
```

```
dhcp:NodeGrp
dhcp:PodGrp
edr:ErrDisRecoverPol
ep:ControlP
ep:LoopProtectP
eqptdiagp:TsOdFabP
eqptdiagp:TsOdLeafP
fabric:AutoGEp
fabric:ExplicitGEp
fabric:FuncP
fabric:HIfPol
fabric:L1IfPol
fabric:L2IfPol
fabric:L2InstPol
fabric:L2PortSecurityPol
fabric:LeCardP
fabric:LeCardPGrp
fabric:LeCards
fabric:LeNodePGrp
fabric:LePortP
fabric:LePortPGrp
fabric:LFPortS
fabric:NodeControl
fabric:OLeafS
fabric:OSpineS
fabric:PodPGrp
fabric:PortBlk
fabric:ProtGEp
fabric:ProtPol
fabric:SFPortS
fabric:SpCardP
fabric:SpCardPGrp
fabric:SpCards
fabric:SpNodePGrp
fabric:SpPortP
fabric:SpPortPGrp
fc:DomP
fc:FabricPol
fc:IfPol
fc:InstPol
file:RemotePath
fvns:McastAddrInstP
fvns:VlanInstP
fvns:VsanInstP
fvns:VxlanInstP
infra:AccBaseGrp
infra:AccBndlGrp
infra:AccBndlPolGrp
infra:AccBndlSubgrp
infra:AccCardP
infra:AccCardPGrp
infra:AccNodePGrp
infra:AccPortGrp
infra:AccPortP
infra:AttEntityP
infra:Cards
infra:ConnFexBlk
infra:ConnFexS
infra:ConnNodeS
infra:DomP
infra:FexBlk
infra:FexBndlGrp
infra:FexGrp
infra:FexP
```

```

infra:FuncP
infra:HConnPortS
infra:HPathS
infra:HPortS
infra:LeafS
infra:NodeBlk
infra:NodeGrp
infra:NodeP
infra:OLeafS
infra:OSpinesS
infra:PodBlk
infra:PodGrp
infra:PodP
infra:PodS
infra:PolGrp
infra:PortBlk
infra:PortP
infra:PortS
infra:PortTrackPol
infra:Profile
infra:SHPathS
infra:SHPortS
infra:SpAccGrp
infra:SpAccPortGrp
infra:SpAccPortP
infra:SpineP
infra:SpineS
isis:DomPol
l2ext:DomP
l2:IfPol
l2:InstPol
l2:PortSecurityPol
l3ext:DomP
lacp:IfPol
lacp:LagPol
lldp:IfPol
lldp:InstPol
mcp:IfPol
mcp:InstPol
mgmt:NodeGrp
mgmt:PodGrp
mon:FabricPol
mon:InfraPol
phys:DomP
psu:InstPol
qos:DppPol
snmp:Pol
span:Dest
span:DestGrp
span:SpanProv
span:SrcGrp
span:SrcTargetShadow
span:SrcTargetShadowBD
span:SrcTargetShadowCtx
span:TaskParam
span:VDest
span:VDestGrp
span:VSpanProv
span:VSrcGrp
stormctrl:IfPol
stp:IfPol
stp:InstPol
stp:MstDomPol
stp:MstRegionPol

```

```

trig:SchedP
vmm:DomP
vpc:InstPol
vpc:KAPol

```

Creating Configuration Zones Using the REST API

Before you begin

This procedure explains how to create a configuration zone using the REST API.

Create a configuration zone using the REST API leaf switch or pod examples below.

Example:

Creating a Config Zone with Leaf Switches

```

<infraInfra>
<infrazoneZoneP name="default">
<infrazoneZone name="Group1" deplMode="disabled">
<infrazoneNodeGrp name="nodeGroup">
<infraNodeBlk name="nodeblk1" from_=101 to_=101/>
<infraNodeBlk name="nodeblk2" from_=103 to_=103/>
</infrazoneNodeGrp>
</infrazoneZone>
<infrazoneZone name="Group2" deplMode="enabled">
<infrazoneNodeGrp name="nodeGroup2">
<infraNodeBlk name="nodeblk" from_=102 to_=102/>
</infrazoneNodeGrp>
</infrazoneZone>
</infrazoneZoneP>
</infraInfra>

```

Example:

Creating a Config Zone with Pods

```

<infraInfra>
  <infrazoneZoneP name="default">
    <infrazoneZone name="testZone" descr="testZone-Description" deplMode="enabled">
      <infrazonePodGrp name="podGroup1">
        <infraPodBlk name="group1" from_=101 to_=101/>
        <infraPodBlk name="group2" from_=103 to_=103/>
      </infrazonePodGrp>
      <infrazonePodGrp name="podGroup2">
        <infraPodBlk name="group" from_=102 to_=102/>
      </infrazonePodGrp>
    </infrazoneZone>
  </infrazoneZoneP>
</infraInfra>

```



CHAPTER 4

Managing Roles, Users, and Signature-Based Transactions

- [Managing APIC Roles and Users](#), on page 79
- [APIC Signature-Based Transactions](#), on page 84

Managing APIC Roles and Users

User Access, Authorization, and Accounting

Application Policy Infrastructure Controller (APIC) policies manage the authentication, authorization, and accounting (AAA) functions of the Cisco Application Centric Infrastructure (ACI) fabric. The combination of user privileges, roles, and domains with access rights inheritance enables administrators to configure AAA functions at the managed object level in a granular fashion. These configurations can be implemented using the REST API, the CLI, or the GUI.



Note There is a known limitation where you cannot have more than 32 characters for the login domain name. In addition, the combined number of characters for the login domain name and the user name cannot exceed 64 characters.

Accounting

ACI fabric accounting is handled by these two managed objects (MO) that are processed by the same mechanism as faults and events:

- The `aaaSessionLR` MO tracks user account login and logout sessions on the APIC and switches, and token refresh. The ACI fabric session alert feature stores information such as the following:
 - Username
 - IP address initiating the session
 - Type (telnet, https, REST etc.)
 - Session time and length

- Token refresh – a user account login event generates a valid active token which is required in order for the user account to exercise its rights in the ACI fabric.



Note Token expiration is independent of login; a user could log out but the token expires according to the duration of the timer value it contains.

- The `aaaModLR` MO tracks the changes users make to objects and when the changes occurred.
- If the AAA server is not pingable, it is marked unavailable and a fault is seen.

Both the `aaaSessionLR` and `aaaModLR` event logs are stored in APIC shards. Once the data exceeds the pre-set storage allocation size, it overwrites records on a first-in first-out basis.



Note In the event of a destructive event such as a disk crash or a fire that destroys an APIC cluster node, the event logs are lost; event logs are not replicated across the cluster.

The `aaaModLR` and `aaaSessionLR` MOs can be queried by class or by distinguished name (DN). A class query provides all the log records for the whole fabric. All `aaaModLR` records for the whole fabric are available from the GUI at the **Fabric > Inventory > POD > History > Audit Log** section, The APIC GUI **History > Audit Log** options enable viewing event logs for a specific object identified in the GUI.

The standard syslog, callhome, REST query, and CLI export mechanisms are fully supported for `aaaModLR` and `aaaSessionLR` MO query data. There is no default policy to export this data.

There are no pre-configured queries in the APIC that report on aggregations of data across a set of objects or for the entire system. A fabric administrator can configure export policies that periodically export `aaaModLR` and `aaaSessionLR` query data to a syslog server. Exported data can be archived periodically and used to generate custom reports from portions of the system or across the entire set of system logs.

Multiple Tenant Support

A core Application Policy Infrastructure Controller (APIC) internal data access control system provides multitenant isolation and prevents information privacy from being compromised across tenants. Read/write restrictions prevent any tenant from seeing any other tenant's configuration, statistics, faults, or event data. Unless the administrator assigns permissions to do so, tenants are restricted from reading fabric configuration, policies, statistics, faults, or events.

User Access: Roles, Privileges, and Security Domains

The APIC provides access according to a user's role through role-based access control (RBAC). An Cisco Application Centric Infrastructure (ACI) fabric user is associated with the following:

- A set of roles
- For each role, a privilege type: no access, read-only, or read-write
- One or more security domain tags that identify the portions of the management information tree (MIT) that a user can access

The ACI fabric manages access privileges at the managed object (MO) level. A privilege is an MO that enables or restricts access to a particular function within the system. For example, fabric-equipment is a privilege bit. This bit is set by the Application Policy Infrastructure Controller (APIC) on all objects that correspond to equipment in the physical fabric.

A role is a collection of privilege bits. For example, because an “admin” role is configured with privilege bits for “fabric-equipment” and “tenant-security,” the “admin” role has access to all objects that correspond to equipment of the fabric and tenant security.

A security domain is a tag associated with a certain subtree in the ACI MIT object hierarchy. For example, the default tenant “common” has a domain tag `common`. Similarly, the special domain tag `all` includes the entire MIT object tree. An administrator can assign custom domain tags to the MIT object hierarchy. For example, an administrator could assign the “solar” domain tag to the tenant named solar. Within the MIT, only certain objects can be tagged as security domains. For example, a tenant can be tagged as a security domain but objects within a tenant cannot.



Note Security Domain password strength parameters can be configured by creating **Custom Conditions** or by selecting **Any Three Conditions** that are provided.

Creating a user and assigning a role to that user does not enable access rights. It is necessary to also assign the user to one or more security domains. By default, the ACI fabric includes two special pre-created domains:

- `All`—allows access to the entire MIT
- `Infra`— allows access to fabric infrastructure objects/subtrees, such as fabric access policies



Note For read operations to the managed objects that a user's credentials do not allow, a "DN/Class Not Found" error is returned, not "DN/Class Unauthorized to read." For write operations to a managed object that a user's credentials do not allow, an HTTP 401 Unauthorized error is returned. In the GUI, actions that a user's credentials do not allow, either they are not presented, or they are grayed out.

A set of predefined managed object classes can be associated with domains. These classes should not have overlapping containment. Examples of classes that support domain association are as follows:

- Layer 2 and Layer 3 network managed objects
- Network profiles (such as physical, Layer 2, Layer 3, management)
- QoS policies

When an object that can be associated with a domain is created, the user must assign domain(s) to the object within the limits of the user's access rights. Domain assignment can be modified at any time.

If a virtual machine management (VMM) domain is tagged as a security domain, the users contained in the security domain can access the correspondingly tagged VMM domain. For example, if a tenant named solar is tagged with the security domain called sun and a VMM domain is also tagged with the security domain called sun, then users in the solar tenant can access the VMM domain according to their access rights.

Configuring a Custom Role Using the REST API

SUMMARY STEPS

1. To configure a custom role, send a POST request with XML as in the following example:

DETAILED STEPS

To configure a custom role, send a POST request with XML as in the following example:

Example:

```
<aaaRolereresetToFactory="no"
priv="aaa,access-connectivity-l1,access-connectivity-l2,access-connectivity-l3,access-connectivity-mgmt,
access-connectivity-util,access-equipment,access-protocol-l1,access-protocol-l2,access-protocol-l3,access-protocol-mgmt,
access-protocol-ops,access-protocol-util,access-qos,fabric-connectivity-l1,fabric-connectivity-l2,
fabric-connectivity-l3,fabric-connectivity-mgmt,fabric-connectivity-util,fabric-equipment,
fabric-protocol-l1,fabric-protocol-l2,fabric-protocol-l3,fabric-protocol-mgmt,fabric-protocol-ops,
fabric-protocol-util,nw-svc-device,nw-svc-devshare,nw-svc-policy,ops,tenant-connectivity-l1,
tenant-connectivity-l2,tenant-connectivity-l3,tenant-connectivity-mgmt,tenant-connectivity-util,
tenant-epg,tenant-ext-connectivity-l1,tenant-ext-connectivity-l2,tenant-ext-connectivity-l3,
tenant-ext-connectivity-mgmt,tenant-ext-connectivity-util,tenant-ext-protocol-l1,tenant-ext-protocol-l2,
tenant-ext-protocol-l3,tenant-ext-protocol-mgmt,tenant-ext-protocol-util,tenant-network-profile,
tenant-protocol-l1,tenant-protocol-l2,tenant-protocol-l3,tenant-protocol-mgmt,tenant-protocol-ops,
tenant-protocol-util,tenant-qos,tenant-security,vmm-connectivity,vmm-ep,vmm-policy,vmm-protocol-ops,
vmm-security" ownerTag="" ownerKey="" name="tenant-admin" dn="uni/userext/role-tenant-admin"
descr=""/>
```

Configuring a Local User

In the initial configuration script, the admin account is configured and the admin is the only user when the system starts. The APIC supports a granular, role-based access control system where user accounts can be created with various roles including non-admin users with fewer privileges.

Configuring a Local User Using the REST API

SUMMARY STEPS

1. Create a local user.

DETAILED STEPS

Create a local user.

Example:

```
URL: https://apic-ip-address/api/node/mo/uni/userext.xml
POST CONTENT:
<aaaUser name="operations" phone="" pwd="<strong_password>" >
  <aaaUserDomain childAction="" descr="" name="all" rn="userdomain-all" status="">
    <aaaUserRole childAction="" descr="" name="Ops" privType="writePriv"/>
  </aaaUserDomain>
</aaaUser>
```

Configuring a Remote User

Instead of configuring local users, you can point the APIC at the centralized enterprise credential datacenter. The APIC supports Lightweight Directory Access Protocol (LDAP), active directory, RADIUS, and TACACS+.



Note When an APIC is in minority (disconnected from the cluster), remote logins can fail because the ACI is a distributed system and the user information is distributed across APICS. Local logins, however, continue to work because they are local to the APIC.

Starting with the 3.1(1) release, **Server Monitoring** can be configured through RADIUS, TACACS+, LDAP, and RSA to determine whether the respective AAA servers are alive or not. Server monitoring feature uses the respective protocol login to check for server aliveness. For example, a LDAP server will use ldap login and a Radius server will use radius login with server monitoring to determine server aliveness.

To configure a remote user authenticated through an external authentication provider, you must meet the following prerequisites:

- The DNS configuration should have already been resolved with the hostname of the RADIUS server.
- You must configure the management subnet.

Configuring a Remote User Using the REST API

SUMMARY STEPS

1. Create a RADIUS provider.
2. Create a login domain.

DETAILED STEPS

Step 1 Create a RADIUS provider.

Example:

```
URL: https://apic-ip-address/api/policymgr/mo/uni/userext/radiusext.xml
POST Content:
<aaaRadiusProvider name="radius-auth-server.org.com" key="test123" />
```

Step 2 Create a login domain.

Example:

```
URL: https://apic-ip-address/api/policymgr/mo/uni/userext.xml
POST Content:
<aaaLoginDomain name="rad"> <aaaDomainAuth realm="radius"/> </aaaLoginDomain>
```

APIC Signature-Based Transactions

About Signature-Based Transactions

The APIC controllers in a Cisco ACI fabric offer different methods to authenticate users.

The primary authentication method uses a username and password and the APIC REST API returns an authentication token that can be used for future access to the APIC. This may be considered insecure in a situation where HTTPS is not available or enabled.

Another form of authentication that is offered utilizes a signature that is calculated for every transaction. The calculation of that signature uses a private key that must be kept secret in a secure location. When the APIC receives a request with a signature rather than a token, the APIC utilizes an X.509 certificate to verify the signature. In signature-based authentication, every transaction to the APIC must have a newly calculated signature. This is not a task that a user should do manually for each transaction. Ideally this function should be utilized by a script or an application that communicates with the APIC. This method is the most secure as it requires an attacker to crack the RSA/DSA key to forge or impersonate the user credentials.



Note Additionally, you must use HTTPS to prevent replay attacks.

Before you can use X.509 certificate-based signatures for authentication, verify that the following pre-requisite tasks are completed:

1. Create an X.509 certificate and private key using OpenSSL or a similar tool.
2. Create a local user on the APIC. (If a local user is already available, this task is optional).
3. Add the X.509 certificate to the local user on the APIC.

Using a Private Key to Calculate a Signature

Before you begin

You must have the following information available:

- HTTP method - GET, POST, DELETE

- REST API URI being requested, including any query options
- For POST requests, the actual payload being sent to the APIC
- The private key used to generate the X.509 certificate for the user
- The distinguished name for the user X.509 certificate on the APIC

Step 1 Concatenate the HTTP method, REST API URI, and payload together in this order and save them to a file.

This concatenated data must be saved to a file for OpenSSL to calculate the signature. In this example, we use a filename of payload.txt. Remember that the private key is in a file called userabc.key.

Example:

GET example:

```
GET http://10.10.10.1/api/class/fvTenant.json?rsp-subtree=children
```

POST example:

```
POST http://10.10.10.1/api/mo/tn-test.json{"fvTenant": {"attributes": {"status": "deleted", "name": "test"}}
```

Step 2 Verify that the payload.txt file contains the correct information.

For example, using the GET example shown in the previous step:

```
GET http://10.10.10.1/api/class/fvTenant.json?rsp-subtree=children
```

Your payload.txt file should contain only the following information:

```
GET/api/class/fvTenant.json?rsp-subtree=children
```

Step 3 Verify that you didn't inadvertently create a new line when you created the payload file.

Example:

```
# cat -e payload.txt
```

Determine if there is a \$ symbol at the end of the output, similar to the following:

```
GET/api/class/fvTenant.json?rsp-subtree=children$
```

If so, then that means that a new line was created when you created the payload file. To prevent creating a new line when generating the payload file, use a command similar to the following:

```
echo -n "GET/api/class/fvTenant.json?rsp-subtree=children" >payload.txt
```

Step 4 Calculate a signature using the private key and the payload file using OpenSSL.

Example:

```
openssl dgst -sha256 -sign userabc.key payload.txt > payload_sig.bin
```

The resulting file has the signature printed on multiple lines.

Step 5 Convert the signature to base64 format:

Example:

```
openssl base64 -A -in payload_sig.bin -out payload_sig.base64
```

Step 6 Strip the signature of the new lines using Bash.

Example:

```
$ tr -d '\n' < payload_sig.base64
P+OTqK0CeAZj17+Gute2R1Ww8OGgtzE0wsLlx8fIXX14V79Z17
Ou8IdJH9CB4W6CEvdICXqkv3KaQszCIC0+Bn07o3qF//BsIplZmYChD6gCX3f7q
IcJGX+R6HAqGeK7k97cNhXlWEoobFPe/oaJtPjOu3tdOjhf/9ujG6Jv6Ro=
```

Note This is the signature that will be sent to the APIC for this specific request. Other requests will require to have their own signatures calculated.

Step 7 Place the signature inside a string to enable the APIC to verify the signature against the payload.

This complete signature is sent to the APIC as a cookie in the header of the request.

Example:

```
APIC-Request-Signature=P+OTqK0CeAZj17+Gute2R1Ww8OGgtzE0wsLlx8f
IXX14V79Z17Ou8IdJH9CB4W6CEvdICXqkv3KaQszCIC0+Bn07o3qF//BsIplZmYChD6gCX3f
7qIcJGX+R6HAqGeK7k97cNhXlWEoobFPe/oaJtPjOu3tdOjhf/9ujG6Jv6Ro=;
APIC-Certificate-Algorithm=v1.0; APIC-Certificate-Fingerprint=fingerprint;
APIC-Certificate-DN=uni/userext/user-userabc/usercert-userabc.crt
```

Note The DN used here must match the DN of the user certified object containing the x509 certificate in the next step.

Step 8 Use the CertSession class in the Python SDK to communicate with an APIC using signatures.

The following script is an example of how to use the CertSession class in the ACI Python SDK to make requests to an APIC using signatures.

Example:

```
#!/usr/bin/env python
# It is assumed the user has the X.509 certificate already added to
# their local user configuration on the APIC
from cobra.mit.session import CertSession
from cobra.mit.access import MoDirectory

def readFile(fileName=None, mode="r"):
    if fileName is None:
        return ""
    fileData = ""
    with open(fileName, mode) as aFile:
        fileData = aFile.read()
    return fileData

pkey = readFile("/tmp/userabc.key")
csession = CertSession("https://ApicIPorHostname/",
                       "uni/userext/user-userabc/usercert-userabc", pkey)

modir = MoDirectory(csession)
resp = modir.lookupByDn('uni/fabric')
print resp.dn
# End of script
```

Note The DN used in the earlier step must match the DN of the user certified object containing the x509 certificate in this step.

Guidelines and Limitations

Follow these guidelines and limitations:

- Local users are supported. Remote AAA users are not supported.
- The APIC GUI does not support the certificate authentication method.
- WebSockets and eventchannels do not work for X.509 requests.
- Certificates signed by a third party are not supported. Use a self-signed certificate.

Creating a Local User and Adding a User Certificate Using the REST API

Create a local user and add a user certificate.

Example:

```
method: POST
url: http://apic/api/node/mo/uni/userext/user-userabc.json
payload:
{
  "aaaUser": {
    "attributes": {
      "name": "userabc",
      "firstName": "Adam",
      "lastName": "BC",
      "phone": "408-525-4766",
      "email": "userabc@cisco.com",
    },
    "children": [{
      "aaaUserCert": {
        "attributes": {
          "name": "userabc.crt",
          "data": "-----BEGIN CERTIFICATE-----\nMIICjjCCAfegAwIBAgIJAMQnBE <snipped
content> ==\n-----END CERTIFICATE-----",
        },
        "children": []
      },
      "aaaUserDomain": {
        "attributes": {
          "name": "all",
        },
        "children": [{
          "aaaUserRole": {
            "attributes": {
              "name": "aaa",
              "privType": "writePriv",
            },
            "children": []
          }
        }, {
          "aaaUserRole": {
            "attributes": {
              "name": "access-admin",
              "privType": "writePriv",
            },
            "children": []
          }
        }, {
          "aaaUserRole": {
            "attributes": {
              "name": "admin",
              "privType": "writePriv",
            }
          }
        }
      ]
    }
  }
}
```

```

        },
        "children": []
    }, {
        "aaaUserRole": {
            "attributes": {
                "name": "fabric-admin",
                "privType": "writePriv",
            },
            "children": []
        }
    }, {
        "aaaUserRole": {
            "attributes": {
                "name": "nw-svc-admin",
                "privType": "writePriv",
            },
            "children": []
        }
    }, {
        "aaaUserRole": {
            "attributes": {
                "name": "ops",
                "privType": "writePriv",
            },
            "children": []
        }
    }, {
        "aaaUserRole": {
            "attributes": {
                "name": "read-all",
                "privType": "writePriv",
            },
            "children": []
        }
    }, {
        "aaaUserRole": {
            "attributes": {
                "name": "tenant-admin",
                "privType": "writePriv",
            },
            "children": []
        }
    }, {
        "aaaUserRole": {
            "attributes": {
                "name": "tenant-ext-admin",
                "privType": "writePriv",
            },
            "children": []
        }
    }, {
        "aaaUserRole": {
            "attributes": {
                "name": "vmm-admin",
                "privType": "writePriv",
            },
            "children": []
        }
    }
    ]
}

```

```
}
```



CHAPTER 5

Common Tenant Tasks

- [Common Tenant Tasks, on page 91](#)

Common Tenant Tasks

Tenants Overview

- A tenant contains policies that enable qualified users domain-based access control. Qualified users can access privileges such as tenant administration and networking administration.
- A user requires read/write privileges for accessing and configuring policies in a domain. A tenant user can have specific privileges into one or more domains.
- In a multitenancy environment, a tenant provides group user access privileges so that resources are isolated from one another (such as for endpoint groups and networking). These privileges also enable different users to manage different tenants.

Tenant Creation

A tenant contains primary elements such as filters, contracts, bridge domains, and application profiles that you can create after you first create a tenant.

Adding a Tenant

A tenant is a policy owner in the virtual fabric. A tenant can be either a private or a shared entity. For example, you can create a securely partitioned private tenant or a tenant with contexts and bridge domains shared by other tenants. A shared type of tenant is typically named `common`, `default`, or `infra`.

In the management information model, a tenant is represented by a managed object (MO) of class `fv:Tenant`. According to the *Cisco APIC Management Information Model Reference*, an object of the `fv:Tenant` class is a child of the policy resolution universe (`uni`) class and has a distinguished name (DN) format of `uni/tn-[name]`.



Note You can only add one tenant at a time.

The following examples show how to add a new tenant named ExampleCorp using XML and JSON.

Example: Using the JSON API to Add a Tenant

To create a new tenant, you must specify the class and sufficient naming information, either in the message body or in the URI.

To create a new tenant using the JSON API, send this HTTP POST message:

```
POST https://apic-ip-address/api/mo/uni.json
```

```
{
  "fvTenant" : {
    "attributes" : {
      "name" : "ExampleCorp"
    }
  }
}
```

Alternatively, you can name the tenant in the URI, as in this example:

```
POST https://apic-ip-address/api/mo/uni/tn-ExampleCorp.json
```

```
{
  "fvTenant" : {
    "attributes" : {
    }
  }
}
```

If a response is requested (by appending `?rsp-subtree=modified` to the POST URI), a successful operation returns the following response body:

```
{
  "imdata" :
  [{
    "fvTenant" : {
      "attributes" : {
        "instanceId" : "0:0",
        "childAction" : "deleteNonPresent",
        "dn" : "uni/tn-ExampleCorp",
        "lcOwn" : "local",
        "name" : "ExampleCorp",
        "replTs" : "never",
        "rn" : "",
        "status" : "created"
      }
    }
  ]
}
```

To delete the tenant, send this HTTP DELETE message:

```
DELETE https://apic-ip-address/api/mo/uni/tn-ExampleCorp.json
```

Alternatively, you can send an HTTP POST message with sufficient naming information and with "status" : "deleted" in the fv:Tenant attributes, as in this example:

```
POST https://apic-ip-address/api/mo/uni.json

{
  "fvTenant" : {
    "attributes" : {
      "name" : "ExampleCorp",
      "status" : "deleted"
    }
  }
}
```

Example: Using the XML API to Add a Tenant

To create a new tenant, you must specify the class and sufficient naming information, either in the message body or in the URI.

To create a new tenant named ExampleCorp using the XML API, send this HTTP POST message:

```
POST https://apic-ip-address/api/mo/uni.xml

<fvTenant name="ExampleCorp"/>
```

Alternatively, you can name the tenant in the URI, as in this example:

```
POST https://apic-ip-address/api/mo/uni/tn-ExampleCorp.xml

<fvTenant />
```

If a response is requested (by appending `?rsp-subtree=modified` to the POST URI), a successful operation returns the following response body:

```
<imdata>
  <fvTenant
    instanceId="0:0"
    childAction="deleteNonPresent"
    dn="uni/tn-ExampleCorp"
    lcOwn="local"
    name="ExampleCorp"
    replTs="never"
    rn=""
    status="created"
  />
</imdata>
```

To delete the tenant, send this HTTP DELETE message:

```
DELETE https://apic-ip-address/api/mo/uni/tn-ExampleCorp.xml
```

Alternatively, you can send an HTTP POST message with sufficient naming information and with `status="deleted"` in the fv:Tenant attributes, as in this example:

Example: Using the XML API to Add a Tenant

```
POST https://apic-ip-address/api/mo/uni.xml  
<fvTenant name="ExampleCorp" status="deleted"/>
```




CHAPTER 6

Managing Layer 2 Networking

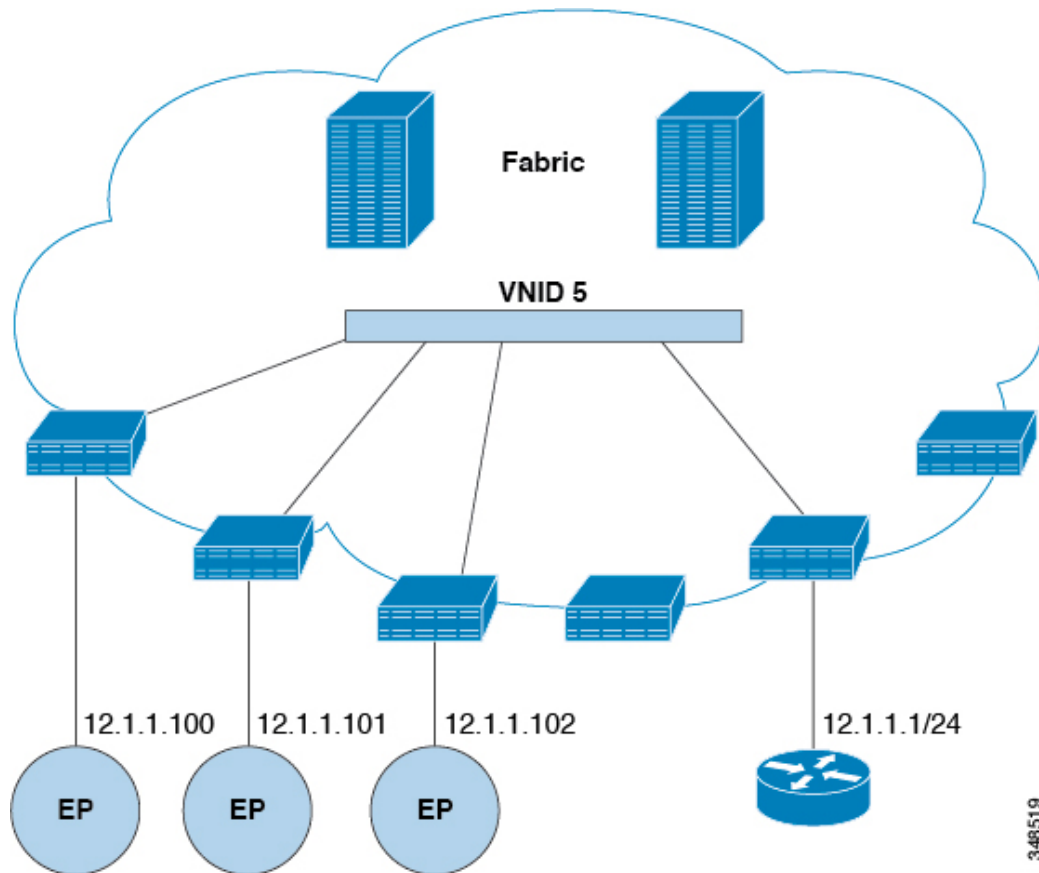
- [Tenant External Bridged Networks, on page 95](#)
- [Ports, on page 97](#)
- [Creating a Port Channel Policy Using the REST API, on page 100](#)

Tenant External Bridged Networks

Bridged Interface to an External Router

As shown in the figure below, when the leaf switch interface is configured as a bridged interface, the default gateway for the tenant VNID is the external router.

Figure 4: Bridged External Router



The ACI fabric is unaware of the presence of the external router and the APIC statically assigns the leaf switch interface to its EPG.

VRF and Bridge Domains

You can create and specify a VRF and a bridge domain for the tenant. The defined bridge domain element subnets reference a corresponding Layer 3 context.

For details about enabling IPv6 Neighbor Discovery see *IPv6 and Neighbor Discovery in Cisco APIC Layer 3 Networking Guide*.

Creating a Tenant, VRF, and Bridge Domain Using the REST API

SUMMARY STEPS

1. Create a tenant.
2. Create a VRF and bridge domain.

DETAILED STEPS

Step 1 Create a tenant.

Example:

```
POST https://apic-ip-address/api/mo/uni.xml
<fvTenant name="ExampleCorp"/>
```

When the POST succeeds, you see the object that you created in the output.

Step 2 Create a VRF and bridge domain.

Note The Gateway Address can be an IPv4 or an IPv6 address. For more about details IPv6 gateway address, see the related KB article, *KB: Creating a Tenant, VRF, and Bridge Domain with IPv6 Neighbor Discovery*.

Example:

URL for POST: `https://apic-ip-address/api/mo/uni/tn-ExampleCorp.xml`

```
<fvTenant name="ExampleCorp">
  <fvCtx name="pvn1"/>
  <fvBD name="bd1">
    <fvRsCtx tnFvCtxName="pvn1"/>
    <fvSubnet ip="10.10.100.1/24"/>
  </fvBD>
</fvTenant>
```

Note If you have a public subnet when you configure the routed outside, you must associate the bridge domain with the outside configuration.

Ports

Statically Deploying an EPG on a Specific Port

This topic provides a typical example of how to statically deploy an EPG on a specific port when using Cisco APIC.

Deploying an EPG on a Specific Port with APIC Using the REST API

Before you begin

The tenant where you deploy the EPG is created.

Deploy an EPG on a specific port.

Example:

```
<fvTenant name="<tenant_name>" dn="uni/tn-test1" >
  <fvCtx name="<network_name>" pcEnfPref="enforced" knwMcastAct="permit"/>
  <fvBD name="<bridge_domain_name>" unkMcastAct="flood" >
    <fvRsCtx tnFvCtxName="<network_name>" />
```

```

</fvBD>
<fvAp name="<application_profile>" >
  <fvAEPg name="<epg_name>" >
    <fvRsPathAtt tDn="topology/pod-1/paths-1017/pathep-[eth1/13]" mode="regular"
instrImedcy="immediate" encap="vlan-20"/>
  </fvAEPg>
</fvAp>
</fvTenant>

```

Creating Domains, Attach Entity Profiles, and VLANs to Deploy an EPG on a Specific Port

This topic provides a typical example of how to create physical domains, Attach Entity Profiles (AEP), and VLANs that are mandatory to deploy an EPG on a specific port.

All endpoint groups (EPGs) require a domain. Interface policy groups must also be associated with Attach Entity Profile (AEP), and the AEP must be associated with a domain, if the AEP and EPG have to be in same domain. Based on the association of EPGs to domains and of interface policy groups to domains, the ports and VLANs that the EPG uses are validated. The following domain types associate with EPGs:

- Application EPGs
- Layer 3 external outside network instance EPGs
- Layer 2 external outside network instance EPGs
- Management EPGs for out-of-band and in-band access

The APIC checks if an EPG is associated with one or more of these types of domains. If the EPG is not associated, the system accepts the configuration but raises a fault. The deployed configuration may not function properly if the domain association is not valid. For example, if the VLAN encapsulation is not valid for use with the EPG, the deployed configuration may not function properly.



Note EPG association with the AEP without static binding does not work in a scenario when you configure the EPG as **Trunk** under the AEP with one end point under the same EPG supporting Tagging and the other end point in the same EPG does not support VLAN tagging. While associating AEP under the EPG, you can configure it as Trunk, Access (Tagged) or Access (Untagged).

Creating AEP, Domains, and VLANs to Deploy an EPG on a Specific Port Using the REST API

Before you begin

- The tenant where you deploy the EPG is already created.
- An EPG is statically deployed on a specific port.

Step 1 Create the interface profile, switch profile and the Attach Entity Profile (AEP).**Example:**

```

<infraInfra>

  <infraNodeP name="<switch_profile_name>" dn="uni/infra/nprof-<switch_profile_name>" >
    <infraLeafS name="SwitchSeletor" descr="" type="range">
      <infraNodeBlk name="nodeBlk1" descr="" to_="1019" from_="1019"/>
    </infraLeafS>
    <infraRsAccPortP tDn="uni/infra/accportprof-<interface_profile_name>"/>
  </infraNodeP>

  <infraAccPortP name="<interface_profile_name>" dn="uni/infra/accportprof-<interface_profile_name>"
  >
    <infraHPortS name="portSelector" type="range">
      <infraRsAccBaseGrp tDn="uni/infra/funcprof/accportgrp-<port_group_name>" fexId="101"/>

      <infraPortBlk name="block2" toPort="13" toCard="1" fromPort="11" fromCard="1"/>
    </infraHPortS>
  </infraAccPortP>

  <infraAccPortGrp name="<port_group_name>" dn="uni/infra/funcprof/accportgrp-<port_group_name>"
  >
    <infraRsAttEntP tDn="uni/infra/attentp-<attach_entity_profile_name>"/>
    <infraRsHIfPol tnFabricHIfPolName="lGHifPol"/>
  </infraAccPortGrp>

  <infraAttEntityP name="<attach_entity_profile_name>"
dn="uni/infra/attentp-<attach_entity_profile_name>" >
    <infraRsDomP tDn="uni/phys-<physical_domain_name>"/>
  </infraAttEntityP>

</infraInfra>

```

Step 2 Create a domain.**Example:**

```

<physDomP name="<physical_domain_name>" dn="uni/phys-<physical_domain_name>">
  <infraRsVlanNs tDn="uni/infra/vlanns-[<vlan_pool_name>]-static"/>
</physDomP>

```

Step 3 Create a VLAN range.**Example:**

```

<fvnsVlanInstP name="<vlan_pool_name>" dn="uni/infra/vlanns-[<vlan_pool_name>]-static"
allocMode="static">
  <fvnsEncapBlk name="" descr="" to="vlan-25" from="vlan-10"/>
</fvnsVlanInstP>

```

Step 4 Associate the EPG with the domain.**Example:**

```

<fvTenant name="<tenant_name>" dn="uni/tn-" >
  <fvAEPg prio="unspecified" name="<epg_name>" matchT="AtleastOne"
dn="uni/tn-test1/ap-AP1/epg-<epg_name>" descr="">
    <fvRsDomAtt tDn="uni/phys-<physical_domain_name>" instrImedcy="immediate"
resImedcy="immediate"/>

```

```
</fvAEPg>
</fvTenant>
```

Creating a Port Channel Policy Using the REST API

The following example REST request creates a Port Channel policy:

```
<lacpLagPol childAction="" ctrl="fast-sel-hot-stdby,graceful-conv,susp-individual"
  descr="" dn="uni/infra/lacplagp-LACP-Active" lcOwn="local" maxLinks="16" minLinks="1"
  modTs="2015-02-24T11:58:36.547-08:00" mode="active" name="LACP-Active" ownerKey=""
  ownerTag="" status="" uid="8131">
  <lacpRtLacpPol childAction="" lcOwn="local" modTs="2015-02-24T14:59:11.154-08:00"
    rn="rtinfraLacpPol-[uni/infra/funcprof/accbundle-ACI-VPC-IPG]" status=""
    tCl="infraAccBndlGrp" tDn="uni/infra/funcprof/accbundle-ACI-VPC-IPG"/>
</lacpLagPol>
```



Note

- To enable symmetric hashing, add `ctrl="symmetric-hash"` to the REST request.
- Symmetric hashing is not supported on the following switches:
 - Cisco Nexus 93128TX
 - Cisco Nexus 9372PX
 - Cisco Nexus 9372PX-E
 - Cisco Nexus 9372TX
 - Cisco Nexus 9372TX-E
 - Cisco Nexus 9396PX
 - Cisco Nexus 9396TX



CHAPTER 7

Managing Layer 3 Networking

This chapter contains the following sections:

- [Configuring External Connectivity Using a Layer 3 Out, on page 101](#)
- [Configuring a Tenant Layer 3 Outside Network Connection Overview, on page 101](#)
- [Configuring Layer 3 Outside for Tenant Networks Using the REST API, on page 102](#)
- [Configuring BGP Max Path, on page 105](#)
- [Configuring AS Path Prepend, on page 105](#)
- [Configuring BFD, on page 106](#)

Configuring External Connectivity Using a Layer 3 Out

This section provides a step-by-step configuration required for the ACI fabric to connect to an external routed network through L3Outs and MP-BGP route reflectors.

This example uses Open Shortest Path First (OSPF) as the routing protocol in an L3Out under the 'mgmt' tenant.

Configuring a Tenant Layer 3 Outside Network Connection Overview

This topic provides a typical example of how to configure a Layer 3 Outside for tenant networks when using Cisco APIC.



Note Cisco ACI does not support IP fragmentation. Therefore, when you configure Layer 3 Outside (L3Out) connections to external routers, or Multi-Pod connections through an Inter-Pod Network (IPN), it is recommended that the interface MTU is set appropriately on both ends of a link. On some platforms, such as Cisco ACI, Cisco NX-OS, and Cisco IOS, the configurable MTU value does not take into account the Ethernet headers (matching IP MTU, and excluding the 14-18 Ethernet header size), while other platforms, such as IOS-XR, include the Ethernet header in the configured MTU value. A configured value of 9000 results in a max IP packet size of 9000 bytes in Cisco ACI, Cisco NX-OS, and Cisco IOS, but results in a max IP packet size of 8986 bytes for an IOS-XR untagged interface.

For the appropriate MTU values for each platform, see the relevant configuration guides.

We highly recommend that you test the MTU using CLI-based commands. For example, on the Cisco NX-OS CLI, use a command such as `ping 1.1.1.1 df-bit packet-size 9000 source-interface ethernet 1/1`.

Configuring Layer 3 Outside for Tenant Networks Using the REST API

The external routed network that is configured in the example can also be extended to support both IPv4 and IPv6. Both IPv4 and IPv6 routes can be advertised to and learned from the external routed network. To configure an L3Out for a tenant network, send a post with XML such as the example.

This example is broken into steps for clarity. For a merged example, see [REST API Example: L3Out, on page 312](#).

Before you begin

- Configure the node, port, functional profile, AEP, and Layer 3 domain.
- Create the external routed domain and associate it to the interface for the L3Out.
- Configure a BGP route reflector policy to propagate the routes within the fabric.

For an XML example of these prerequisites, see [REST API Example: L3Out Prerequisites, on page 311](#).

Step 1 Configure the tenant, VRF, and bridge domain.

This example configures tenant `t1` with VRF `v1` and bridge domain `bd1`. The tenant, VRF, and BD are not yet deployed.

Example:

```
<fvTenant name="t1">
  <fvCtx name="v1"/>
  <fvBD name="bd1">
    <fvRsCtx tnFvCtxName="v1"/>
    <fvSubnet ip="44.44.44.1/24" scope="public"/>
    <fvRsBDToOut tnL3extOutName="l3out1"/>
  </fvBD/>
</fvTenant>
```

Step 2 Configure an application profile and application EPG.

This example configures application profile `app1` (on node 101), EPG `epg1`, and associates the EPG with `bd1` and the contract `httpCtrct`, as the consumer.

Example:

```
<fvAp name="app1">
  <fvAEPg name="epg1">
    <fvRsDomAtt instrImedcy="immediate" tDn="uni/phys-dom1"/>
    <fvRsBd tnFvBDName="bd1" />
    <fvRsPathAtt encap="vlan-2011" instrImedcy="immediate" mode="regular"
tDn="topology/pod-1/paths-101/pathep-[eth1/3]"/>
    <fvRsCons tnVzBrCPName="httpCtrct"/>
  </fvAEPg>
</fvAp>
```

Step 3 Configure the node and interface.

This example configures VRF `v1` on node 103 (the border leaf switch), with the node profile, `nodep1`, and router ID `11.11.11.103`. It also configures interface `eth1/3` as a routed interface (Layer 3 port), with IP address `12.12.12.1/24` and Layer 3 domain `dom1`.

Example:

```
<l3extOut name="l3out1">
  <l3extRsEctx tnFvCtxName="v1"/>
  <l3extLNodeP name="nodep1">
    <l3extRsNodeL3OutAtt rtrId="11.11.11.103" tDn="topology/pod-1/node-103"/>
    <l3extLIIfP name="ifp1"/>
    <l3extRsPathL3OutAtt addr="12.12.12.3/24" ifInstT="l3-port"
tDn="topology/pod-1/paths-103/pathep-[eth1/3]"/>
    </l3extLIIfP>
  </l3extLNodeP>
  <l3extRsL3DomAtt tDn="uni/l3dom-dom1"/>
</l3extOut>
```

Step 4 Configure the routing protocol.

This example configures BGP as the primary routing protocol, with a BGP peer with the IP address, `15.15.15.2` and ASN `100`.

Example:

```
<l3extOut name="l3out1">
  <l3extLNodeP name="nodep1">
    <bgpPeerP addr="15.15.15.2">
      <bgpAsP asn="100"/>
    </bgpPeerP>
  </l3extLNodeP>
  <bgpExtP/>
</l3extOut>
```

Step 5 Configure the connectivity routing protocol.

This example configures OSPF as the communication protocol, with regular area ID `0.0.0.0`.

Example:

```
<l3extOut name="l3out1">
  <ospfExtP areaId="0.0.0.0" areaType="regular"/>
  <l3extLNodeP name="nodep1">
    <l3extLIIfP name="ifp1">
      <ospfIIfP/>
    </l3extLIIfP>
  </l3extLNodeP>
</l3extOut>
```

Step 6 Configure the external EPG.

This example configures the network 20.20.20.0/24 as external network `extnw1`. It also associates `extnw1` with the route control profile `rp1` and the contract `httpCtrct`, as the provider.

Example:

```
<l3extOut name="l3out1">
  <l3extInstP name="extnw1">
    <l3extSubnet ip="20.20.20.0/24" scope="import-security"/>
    <fvRsProv tnVzBrCPName="httpCtrct"/>
  </l3extInstP>
</l3extOut>
```

Step 7 Optional. Configure a route map.

This example configures a route map for the BGP peer in the outbound direction. The route map is applied for routes that match a destination of 200.3.2.0/24. Also, on a successful match (if the route matches this range) the route AS PATH attribute is updated to 200 and 100.

Example:

```
<fvTenant name="t1">
  <rtctrlSubjP name="match-rule1">
    <rtctrlMatchRtDest ip="200.3.2.0/24"/>
  </rtctrlSubjP>
  <l3extOut name="l3out1">
    <rtctrlProfile name="rp1">
      <rtctrlCtxP name="ctxp1" action="permit" order="0">
        <rtctrlScope>
          <rtctrlRsScopeToAttrP tnRtctrlAttrPName="attrp1"/>
        </rtctrlScope>
        <rtctrlRsCtxPToSubjP tnRtctrlSubjPName="match-rule1"/>
      </rtctrlCtxP>
    </rtctrlProfile>
    <l3extInstP name="extnw1">
      <l3extSubnet ip="20.20.20.0/24" scope="import-security"/>
      <l3extRsInstPToProfile direction='export' tnRtctrlProfileName="rp1"/>
      <fvRsProv tnVzBrCPName="httpCtrct"/>
    </l3extInstP>
  </l3extOut>
</fvTenant>
```

Step 8 This example creates filters and contracts to enable the EPGs to communicate. The external EPG and the application EPG are already associated with the contract `httpCtrct` as provider and consumer respectively. The scope of the contract (where it is applied) can be within the application profile, the tenant, the VRF, or it can be used globally (throughout the fabric). In this example, the scope is the VRF (`context`).**Example:**

```
<vzFilter name="http-filter">
  <vzEntry name="http-e" etherT="ip" prot="tcp"/>
</vzFilter>
<vzBrCP name="httpCtrct" scope="context">
  <vzSubj name="subj1">
    <vzRsSubjFiltAtt tnVzFilterName="http-filter"/>
  </vzSubj>
</vzBrCP>
```

Step 9 Configure Advertise Host Routes.**Example:**

```
"<fvBD dn="uni/tn-t1/BD-b100" hostBasedRouting="yes"/>"
```

Configuring BGP Max Path

The following feature enables you to add the maximum number of paths to the route table to enable equal cost, multipath load balancing.

Configuring BGP Max Path Using the REST API

This following example provides information on how to configure the BGP Max Path feature using the REST API:

```
<fvTenant descr="" dn="uni/tn-t1" name="t1">
  <fvCtx name="v1">
    <fvRsCtxToBgpCtxAfPol af="ipv4-ucast" tnBgpCtxAfPolName="bgpCtxPol1"/>
  </fvCtx>
  <bgpCtxAfPol name="bgpCtxPol1" maxEcmp="8" maxEcmpIbgp="4"/>
</fvTenant>
```

Configuring AS Path Prepend

A BGP peer can influence the best-path selection by a remote peer by increasing the length of the AS-Path attribute. AS-Path Prepend provides a mechanism that can be used to increase the length of the AS-Path attribute by prepending a specified number of AS numbers to it.

AS-Path prepending can only be applied in the outbound direction using route-maps. AS Path prepending does not work in iBGP sessions.

The AS Path Prepend feature enables modification as follows:

Prepend	Appends the specified AS number to the AS path of the route matched by the route map. Note <ul style="list-style-type: none"> You can configure more than one AS number. 4 byte AS numbers are supported. You can prepend a total 32 AS numbers. You must specify the order in which the AS Number is inserted into the AS Path attribute.
Prepend-last-as	Prepends the last AS numbers to the AS path with a range between 1 and 10.

The following table describes the selection criteria for implementation of AS Path Prepend:

Prepend	1	Prepend the specified AS number.
Prepend-last-as	2	Prepend the last AS numbers to the AS path.

DEFAULT	Prepend(1)	Prepend the specified AS number.
---------	------------	----------------------------------

Configuring AS Path Prepend Using the REST API

The following example provides information on how to configure the AS Path Prepend feature using the REST API:

```
<?xml version="1.0" encoding="UTF-8"?>
<fvTenant name="coke">
  <rtctrlAttrP name="attrp1">
    <rtctrlSetASPath criteria="prepend">
      <rtctrlSetASPathASN asn="100" order="1"/>
      <rtctrlSetASPathASN asn="200" order="10"/>
      <rtctrlSetASPathASN asn="300" order="5"/>
    </rtctrlSetASPath/>
    <rtctrlSetASPath criteria="prepend-last-as" lastnum="9" />
  </rtctrlAttrP>

  <l3extOut name="out1">
    <rtctrlProfile name="rpl">
      <rtctrlCtxP name="ctxp1" order="1">
        <rtctrlScope>
          <rtctrlRsScopeToAttrP tnRtctrlAttrPName="attrp1"/>
        </rtctrlScope>
      </rtctrlCtxP>
    </rtctrlProfile>
  </l3extOut>
</fvTenant>
```

Configuring BFD

Configuring BFD Globally Using the REST API

The following REST API shows the global configuration for bidirectional forwarding detection (BFD):

Example:

```
<polUni>
  <infraInfra>
    <bfdIpv4InstPol name="default" echoSrcAddr="1.2.3.4" slowIntvl="1000" minTxIntvl="150"
minRxIntvl="250" detectMult="5" echoRxIntvl="200"/>
    <bfdIpv6InstPol name="default" echoSrcAddr="34::1/64" slowIntvl="1000" minTxIntvl="150"
minRxIntvl="250" detectMult="5" echoRxIntvl="200"/>
  </infraInfra>
</polUni>
```

Configuring BFD Interface Override Using the REST API

The following REST API shows the interface override configuration for bidirectional forwarding detection (BFD):

Example:

```
<fvTenant name="ExampleCorp">
  <bfdIfPol name="bfdIfPol" minTxIntvl="400" minRxIntvl="400" detectMult="5" echoRxIntvl="400"
  echoAdminSt="disabled"/>
  <l3extOut name="l3-out">
    <l3extLNodeP name="leaf1">
      <l3extRsNodeL3OutAtt tDn="topology/pod-1/node-101" rtrId="2.2.2.2"/>
      <l3extLIfP name='portIpv4'>
        <l3extRsPathL3OutAtt tDn="topology/pod-1/paths-101/pathep-[eth1/11]" ifInstT='l3-port'
        addr="10.0.0.1/24" mtu="1500"/>
        <bfdIfP type="sha1" key="password">
          <bfdRsIfPol tnBfdIfPolName='bfdIfPol' />
        </bfdIfP>
      </l3extLIfP>
    </l3extLNodeP>
  </l3extOut>
</fvTenant>
```

Configuring BFD Consumer Protocols Using the REST API

Step 1 The following example shows the interface configuration for bidirectional forwarding detection (BFD):

Example:

```
<fvTenant name="ExampleCorp">
  <bfdIfPol name="bfdIfPol" minTxIntvl="400" minRxIntvl="400" detectMult="5" echoRxIntvl="400"
  echoAdminSt="disabled"/>
  <l3extOut name="l3-out">
    <l3extLNodeP name="leaf1">
      <l3extRsNodeL3OutAtt tDn="topology/pod-1/node-101" rtrId="2.2.2.2"/>
      <l3extLIfP name='portIpv4'>
        <l3extRsPathL3OutAtt tDn="topology/pod-1/paths-101/pathep-[eth1/11]" ifInstT='l3-port'
        addr="10.0.0.1/24" mtu="1500"/>
        <bfdIfP type="sha1" key="password">
          <bfdRsIfPol tnBfdIfPolName='bfdIfPol' />
        </bfdIfP>
      </l3extLIfP>
    </l3extLNodeP>
  </l3extOut>
</fvTenant>
```

Step 2 The following example shows the interface configuration for enabling BFD on OSPF and EIGRP:

Example:

BFD on leaf switch

```
<fvTenant name="ExampleCorp">
```

```

    <ospfIfPol name="ospf_intf_pol" cost="10" ctrl="bfd"/>
    <eigrpIfPol ctrl="nh-self,split-horizon,bfd" dn="uni/tn-Coke/eigrpIfPol-eigrp_if_default"
  </fvTenant>

```

Example:

BFD on spine switch

```

<l3extLNodeP name="bSpine">
  <l3extRsNodeL3OutAtt tDn="topology/pod-1/node-103" rtrId="192.3.1.8">
    <l3extLoopBackIfP addr="10.10.3.1" />
    <l3extInfraNodeP fabricExtCtrlPeering="false" />
  </l3extRsNodeL3OutAtt>
  <l3extLIIfP name='portIf'>
    <l3extRsPathL3OutAtt tDn="topology/pod-1/paths-103/pathep-[eth5/10]" encap='vlan-4'
  ifInstT='sub-interface' addr="20.3.10.1/24"/>
    <ospfIfP>
      <ospfRsIfPol tnOspfIfPolName='ospf_intf_pol' />
    </ospfIfP>
    <bfdIfP name="test" type="sha1" key="hello" status="created,modified">
      <bfdRsIfPol tnBfdIfPolName='default' status="created,modified"/>
    </bfdIfP>
  </l3extLIIfP>
</l3extLNodeP>

```

Step 3 The following example shows the interface configuration for enabling BFD on BGP:

Example:

```

<fvTenant name="ExampleCorp">
  <l3extOut name="l3-out">
    <l3extLNodeP name="leaf1">
      <l3extRsNodeL3OutAtt tDn="topology/pod-1/node-101" rtrId="2.2.2.2"/>
      <l3extLIIfP name='portIpv4'>
        <l3extRsPathL3OutAtt tDn="topology/pod-1/paths-101/pathep-[eth1/11]" ifInstT='l3-port'
      addr="10.0.0.1/24" mtu="1500">
          <bgpPeerP addr="4.4.4.4/24" allowedSelfAsCnt="3" ctrl="bfd" descr="" name=""
        peerCtrl="" ttl="1">
            <bgpRsPeerPfxPol tnBgpPeerPfxPolName="" />
            <bgpAsP asn="3" descr="" name="" />
          </bgpPeerP>
        </l3extRsPathL3OutAtt>
      </l3extLIIfP>
    </l3extLNodeP>
  </l3extOut>
</fvTenant>

```

Step 4 The following example shows the interface configuration for enabling BFD on Static Routes:

Example:

BFD on leaf switch

```

<fvTenant name="ExampleCorp">
  <l3extOut name="l3-out">
    <l3extLNodeP name="leaf1">
      <l3extRsNodeL3OutAtt tDn="topology/pod-1/node-101" rtrId="2.2.2.2">
        <ipRouteP ip="192.168.3.4" rtCtrl="bfd">
          <ipNexthopP nhAddr="192.168.62.2"/>
        </ipRouteP>
      </l3extRsNodeL3OutAtt>
    </l3extLNodeP>
  </l3extOut>
</fvTenant>

```

```

        </ipRouteP>
    </l3extRsNodeL3OutAtt>
    <l3extLIfP name='portIpv4'>
        <l3extRsPathL3OutAtt tDn="topology/pod-1/paths-101/pathep-[eth1/3]" ifInstT='l3-port'
        addr="10.10.10.2/24" mtu="1500" status="created,modified" />
    </l3extLIfP>

    </l3extLNodeP>

</l3extOut>
</fvTenant>

```

Example:**BFD on spine switch**

```

<l3extLNodeP name="bSpine">

    <l3extRsNodeL3OutAtt tDn="topology/pod-1/node-103" rtrId="192.3.1.8">
        <ipRouteP ip="0.0.0.0" rtCtrl="bfd">
            <ipNextHopP nhAddr="192.168.62.2"/>
        </ipRouteP>
    </l3extRsNodeL3OutAtt>

    <l3extLIfP name='portIf'>
        <l3extRsPathL3OutAtt tDn="topology/pod-1/paths-103/pathep-[eth5/10]" encap='vlan-4'
        ifInstT='sub-interface' addr="20.3.10.1/24"/>

        <bfdIfP name="test" type="sha1" key="hello" status="created,modified">
            <bfdRsIfPol tnBfdIfPolName='default' status="created,modified"/>
        </bfdIfP>
    </l3extLIfP>

</l3extLNodeP>

```

Step 5 The following example shows the interface configuration for enabling BFD on IS-IS:

Example:

```

<fabricInst>
    <l3IfPol name="testL3IfPol" bfdIsis="enabled"/>
    <fabricLeafP name="LeNode" >
        <fabricRsLePortP tDn="uni/fabric/leportp-leaf_profile" />
    <fabricLeafS name="spsw" type="range">
    <fabricNodeBlk name="node101" to_"102" from_"101" />
</fabricLeafS>
    </fabricLeafP>

    <fabricSpineP name="SpNode" >
    <fabricRsSpPortP tDn="uni/fabric/sportp-spine_profile" />
    <fabricSpineS name="spsw" type="range">
        <fabricNodeBlk name="node103" to_"103" from_"103" />
    </fabricSpineS>
    </fabricSpineP>

    <fabricLePortP name="leaf_profile">
    <fabricLFPortS name="leafIf" type="range">
    <fabricPortBlk name="spBlk" fromCard="1" fromPort="49" toCard="1" toPort="49" />
        <fabricRsLePortPGrp tDn="uni/fabric/funcprof/leportgrp-LeTestPGrp" />
    </fabricLFPortS>
    </fabricLePortP>

    <fabricSpPortP name="spine_profile">
    <fabricSFPortS name="spineIf" type="range">

```

```
<fabricPortBlk name="spBlk" fromCard="5" fromPort="1" toCard="5" toPort="2" />
<fabricRsSpPortPGrp tDn="uni/fabric/funcprof/spportgrp-SpTestPGrp" />
</fabricSFPorts>
</fabricSpPortP>

<fabricFuncP>
  <fabricLePortPGrp name = "LeTestPGrp">
<fabricRsL3IfPol tnL3IfPolName="testL3IfPol"/>
  </fabricLePortPGrp>

  <fabricSpPortPGrp name = "SpTestPGrp">
<fabricRsL3IfPol tnL3IfPolName="testL3IfPol"/>
  </fabricSpPortPGrp>

</fabricFuncP>

</fabricInst>
```



CHAPTER 8

Monitoring Using the REST API

- [About Monitoring Using the REST API, on page 111](#)
- [APIC, on page 111](#)
- [Fabric, on page 113](#)
- [Switches, on page 114](#)
- [External Monitoring, on page 117](#)

About Monitoring Using the REST API

Monitoring APIC Using the REST API

Proactive monitoring is an important piece of the network administrator's job but is often neglected because solving urgent problems in the network usually takes priority. However, the Application Policy Infrastructure Controller (APIC) will save network administrators time and frustration because it makes it easy to gather statistics and perform analyses. Because statistics are gathered automatically and policies are used and can be re-used in other places, human effort and error are minimized.

The following examples using the REST API can be used to drill into APIC fabric and switch components.

APIC

Monitoring APIC CPU and Memory Usage Using the REST API

The easiest way to quickly verify the health of the controllers is the APIC. Controllers provide information regarding the current status of CPU and memory utilization by creating instances of the *procEntity* class. The *procEntity* is a container of processes in the system. This object holds detailed information about various processes running on the APIC. The *procEntity* objects contain the following useful properties:

- *cpuPct*—CPU utilization
- *maxMemAlloc*—The maximum memory allocated for the system
- *memFree*—The maximum amount of available memory for the system

SUMMARY STEPS

1. Retrieve information about CPU and memory usage using the following REST API call:

DETAILED STEPS

Retrieve information about CPU and memory usage using the following REST API call:

Example:

```
https://apic-ip-address/api/node/class/procEntity.xml?
```

Monitoring APIC Disk Utilization Using the REST API

There are several disks and file systems present on the APIC. The REST API provides ready access to disk space utilization of all partitions on the system and can be used for monitoring this information.

Monitor the disk and file systems on the APIC, by sending a REST API post, such as the following:

Example:

```
https://apic-ip-address/api/node/class/eqptStorage.xml?
```

Monitoring Physical Interface Statistics and Link State Using the REST API

You can use the REST API interface to poll for interface statistics. Several counters are available (for example, RX/TX, input/output / duplex, 30 second rates, 5 minute rate, unicast packets, multicast packets). Using the parent managed object, the children can be derived from it. To do this, you must have a good understanding of the object model and be able to navigate through the model to obtain the information desired using the example below.

Step 1 Use the following base API call to get physical interface statistics:

Example:

```
https://apic-ip-address/api/node/mo/topology/pod-1/node-101/sys/phys-[eth1/1].json
```

Step 2 To determine the total ingress bytes on Leaf 101 port Eth1/1, you can issue the following API call:

Example:

```
/topology/pod-1/node-101/sys/phys-[eth1/1].json
```

Step 3 Visore allows you to dig deeper into the hierarchical tree. From the prior command, the operator can see children of the interface object, such as ingress and egress bytes. The child objects include the following:

Example:

```
/topology/pod-1/node-101/sys/phys-[eth1/1]/dbgEtherStats
```

Fabric

Monitoring LLDP and CDP Neighbor Status Using the REST API

The APIC enables you to determine all LLDP or CDP neighbors in a fabric, using the REST API.

Step 1 To determine the LLDP neighbors, send a POST such as the following:

Example:

```
https://apic-ip-address/api/node/mo/topology/pod-1/node-101/sys/lldp/inst/if-[eth1/1]
```

Step 2 To determine the CDP neighbors, send a POST such as the following:

Example:

```
https://apic-ip-address/api/node/mo/topology/pod-1/node-101/sys/cdp/inst/if-[eth1/1]
```

Monitoring Physical and Bond Interfaces Using the REST API

The APIC uses a bonded interface that is typically dual-homed to two leaf switches for connectivity to the Cisco ACI fabric. It also can use a bonded interface that can be dual-homed to the out-of-band management network.

- *Bond0* is the bond interface used to connect to the fabric itself (to connect to leaf switches that connect into the fabric).
- *Bond1* is the bond interface used to connect to the out-of-band segment (to connect to an OOB segment that allows setup of the APIC itself).

The bond interfaces rely on underlying physical interfaces. It is important to note that the REST API provides link information for both the physical and logical bond interfaces.

Collect information about both the bond interfaces by sending a POST request such as the following example:

Example:

```
https://apic-ip-address/api/node/mo/topology/pod-1/node-1/sys.json?querytarget=subtree&target-subtree-class=l3EncRtdIf
```

Monitoring EPG-Level Statistics Using the REST API

To monitor network-related information for an application, you can investigate the aggregate amount of traffic to a specific tier. For example, you can monitor the amount of traffic to the web tier of a given EPG application with the REST API.

To monitor the traffic for a new project for the epg-web-epg, send a POST request such as the following example:

Example:

```
https://apic-ip-address/api/node/mo/uni/tn-newproject/ap-appl/epg-web-epg.xml?querytarget=
self&rsp-subtree-include=stats
```

Switches

Monitoring Switch CPU Utilization Using the REST API

Spine and leaf switch CPU utilization can be monitored using the following classes, based on the desired timescale and granularity:

- **proc:SysCPU5min**—A class that represents the most current statistics for system CPU in a 5-minute sampling interval. This class updates every 10 seconds.
- **proc:SysCPU15min**—A class that represents the most current statistics for system CPU in a 15-minute sampling interval. This class updates every 5 minutes.
- **proc:SysCPU1h**—A class that represents the most current statistics for system CPU in a 1-hour sampling interval. This class updates every 15 minutes.
- **proc:SysCPU1d**—A class that represents the most current statistics for system CPU in a 1-day sampling interval. This class updates every hour.
- **proc:SysCPU1w**—A class that represents the most current statistics for system CPU in a 1-week sampling interval. This class updates every day.
- **proc:SysCPU1mo**—A class that represents the most current statistics for system CPU in a 1-month sampling interval. This class updates every day.
- **proc:SysCPU1qtr**—A class that represents the most current statistics for system CPU in a 1-quarter sampling interval. This class updates every day.
- **proc:SysCPU1year**—A class that represents the most current statistics for system CPU in a 1-year sampling interval. This class updates every day.

The following example shows how to use these classes for monitoring:

To view the average CPU utilization of all of the fabric switches over the last day, use XML such as in the following example:

Example:

```
https://apic-ip-address//api/node/class/procSysCPU1d.xml?
```

Monitoring Switch Fan Status Using the REST API

The following REST API call(s) and their child objects can be used to monitor the state of the fans on a leaf switch (note that there are 3 slots on this particular switch).

To retrieve the status of the fan trays on the leaf switches, use XML such as the following example:

Example:

```
https://apic-ip-address/api/node/mo/topology/pod-1/node-101/sys/ch/ftslot-1
https://apic-ip-address/api/node/mo/topology/pod-1/node-101/sys/ch/ftslot-2
https://apic-ip-address/api/node/mo/topology/pod-1/node-101/sys/ch/ftslot-3
```

Monitoring Switch Memory Utilization Using the REST API

Spine and leaf switch memory utilization can be monitored using the following classes, based on the desired timescale and granularity:

- **proc:SysMem5min**—A class that represents the most current statistics for system memory in a 5-minute sampling interval. This class updates every 10 seconds.
- **proc:SysMem15min**—A class that represents the most current statistics for system memory in a 15-minute sampling interval. This class updates every 5 minutes.
- **proc:SysMem1h**—A class that represents the most current statistics for system memory in a 1-hour sampling interval. This class updates every 15 minutes.
- **proc:SysMem1d**—A class that represents the most current statistics for system memory in a 1-day sampling interval. This class updates every hour.
- **proc:SysMem1w**—A class that represents the most current statistics for system memory in a 1-week sampling interval. This class updates every day.
- **proc:SysMem1mo**—A class that represents the most current statistics for system memory in a 1-month sampling interval. This class updates every day.
- **proc:SysMem1qtr**—A class that represents the most current statistics for system memory in a 1-quarter sampling interval. This class updates every day.
- **proc:SysMem1year**—A class that represents the most current statistics for system memory in a 1-year sampling interval. This class updates every day.

The following example shows how to use one of the classes:

To monitor memory over the last day, use the following REST call:

Example:

```
https://apic-ip-address/api/node/class/procSysMem1d.xml?
```

Monitoring Switch Module Status Using the REST API

Even though the leaves are considered fixed switches, they have a supervisor component that refers to the CPU complex. From a forwarding perspective, there are two data-plane components: the NFE (Network Forwarding Engine) ASIC, which provides the front panel ports; and the ALE or ALE2 (Application Leaf Engine) ASIC—depending on the generation of switch hardware—which provides uplink connectivity to the spines. The following REST API example can be used to determine the status of the modules in the switch:

To monitor the state of the supervisor and the module, use a REST API call such as the following:

Example:

```
https://apic-ip-address/api/node/mo/topology/pod-1/node-101/sys/ch/supslot-1/sup
https://apic-ip-address/api/node/mo/topology/pod-1/node-101/sys/ch/lcslot-1/lc
```

Monitoring Switch Power Supply Status Using the REST API

You can use the REST API to retrieve the status of the power supplies on the leaf switches.

To monitor the state of the power supplies on a leaf switch, use XML such as the following example:

Example:

```
https://apic-ip-address/api/node/mo/topology/pod-1/node-101/sys/ch/psuslot-1
https://apic-ip-address/api/node/mo/topology/pod-1/node-101/sys/ch/psuslot-2
```

Note that there are 2 power supplies on this particular switch.

Monitoring Switch Inventory Using the REST API

You can use the REST API to retrieve switch hardware information such as the model and serial numbers.

To retrieve switch hardware information, use the REST API as shown in the following example:

Example:

```
https://apic-ip-address/api/node/mo/topology/pod-1.json?query-target=children&target-subtree-class=fabricNode
```

External Monitoring

Smart Callhome

About Smart Callhome

Smart Callhome provides an email-based notification for critical system policies in a similar way as Callhome. However, Smart Callhome collects a more specific selection of faults to deliver in email messages.



Note Smart Callhome only collects and delivers faults.

The fault triggers that are typical of the Smart Callhome feature correspond to the kind of events that threaten to disrupt your network. Examples are:

- Temperature Faults: The temperature of a sensor exceeds a threshold.
- Fan/ Power Supply Faults: A fan or power supply unit goes offline.
- Disk Utilization Faults: The disk usage of a device exceeds a threshold.

Smart Callhome collects faults and emails them to a network support engineer, a Network Operations Center, or to Cisco Smart Callhome services to generate a case with the Technical Assistance Center (TAC).

Creating a Smart Callhome Destination Group Using the REST API

SUMMARY STEPS

1. Create a Smart Callhome destination group.

DETAILED STEPS

Create a Smart Callhome destination group.

Example:

POST <https://192.168.1.141/api/node/mo/uni/fabric.json>

```
{
  "callhomeSmartGroup": {
    "attributes": {
      "name": "<destination-group-name>",
      "descr": "<description>"
    },
    "children": [
      {
        "callhomeSmartDest": {
          "attributes": {
            "name": "<destination-name>",
            "email": "<email-address>",
            "format="xml"
          },
        },
      }
    ]
  }
}
```


(ModLR) such as the addition of a new user or a password change. Additionally, all configuration changes are logged and include the user ID and time stamp.

Creating a TACACS External Logging Destination Group Using the REST API

Create a TACACS destination group.

Example:

POST `https://<apic-name>/api/node/mo/uni/fabric/tacacsgroup-<groupname>.json`

```
{
  "tacacsGroup": {
    "attributes": {
      "dn": "uni/fabric/tacacsgroup-<groupname>",
      "name": "<groupname>",
      "rn": "tacacsgroup-<groupname>",
      "status": "created"
    },
    "children": [{
      "tacacsTacacsDest": {
        "attributes": {
          "dn": "uni/fabric/tacacsgroup-<groupname>/tacacsdest-<dest-name>-port-<portno>",
          "host": "<dest-name>",
          "rn": "tacacsdest-<dest-name>-port-<portno>",
          "key": "<server secret>",
          "status": "created"
        },
        "children": [{
          "fileRsARemoteHostToEpg": {
            "attributes": {
              "tDn": "uni/tn-mgmt/mgmt-default/oob-default",
              "status": "created"
            },
            "children": []
          }
        ]
      }
    ]
  }
}
```

Creating a TACACS External Logging Source Using the REST API

Create a TACACS source.

Example:

POST `https://<apic-name>/api/node/mo/uni/fabric/moncommon/tacacssrc-<src-name>.json`

```
{
  "tacacsSrc": {
    "attributes": {
      "dn": "uni/fabric/moncommon/tacacssrc-<src-name>",
      "incl": "audits, faults",
      "name": "aaa",
      "rn": "tacacssrc-<src-name>",
      "status": "created",
    }
  }
}
```

```
    "incl": "audit, session"
  },
  "children": [
    {
      "tacacsRsDestGroup": {
        "attributes": {
          "tDn": "uni/fabric/tacacsgroup-<>groupname>",
          "status": "created"
        },
        "children": []
      }
    }
  ]
}
```



CHAPTER 9

Troubleshooting Using the REST API

- [Collecting and Exporting Technical Support Information, on page 121](#)
- [Troubleshooting Using Atomic Counters, on page 122](#)
- [Troubleshooting Using Faults, on page 129](#)
- [Statistics, on page 131](#)
- [Recovering a Disconnected Leaf, on page 132](#)
- [Troubleshooting Contracts and Taboo Contracts with Permit and Deny Logging, on page 133](#)
- [Troubleshooting Using Digital Optical Monitoring Statistics, on page 135](#)
- [Troubleshooting Using Port Tracking, on page 135](#)
- [Removing Unwanted _ui_ Objects, on page 137](#)
- [Troubleshooting Using Contract Permit and Deny Logs, on page 137](#)

Collecting and Exporting Technical Support Information

About Exporting Files

An administrator can configure export policies in the APIC to export statistics, technical support collections, faults and events, to process core files and debug data from the fabric (the APIC as well as the switch) to any external host. The exports can be in a variety of formats, including XML, JSON, web sockets, secure copy protocol (SCP), or HTTP. You can subscribe to exports in streaming, periodic, or on-demand formats.

An administrator can configure policy details such as the transfer protocol, compression algorithm, and frequency of transfer. Policies can be configured by users who are authenticated using AAA. A security mechanism for the actual transfer is based on a username and password. Internally, a policy element handles the triggering of data.

Sending an On-Demand Tech Support File Using the REST API

Step 1 Set the remote destination for a technical support file using the REST API, by sending a POST with XML such as the following example:

Example:

```
<fileRemotePath userName="" remotePort="22" remotePath="" protocol="sftp" name="ToSupport"
host="192.168.200.2"
```

```

dn="uni/fabric/path-ToSupport" descr="">
<fileRsARemoteHostToEpg tDn="uni/tn-mgmt/mgmtp-default/oob-default"/>
</fileRemotePath>

```

Step 2 Generate an on-demand technical support file using the REST API by sending a POST with XML such as the following:

Example:

```

<dbgexpTechSupOnD upgradeLogs="no" startTime="unspecified" name="Tech_Support_9-20-16"
  exportToController="no" endTime="unspecified" dn="uni/fabric/tsod-Tech_Support_9-20-16" descr=""
  compression="gzip" category="forwarding" adminSt="untriggered">
  <dbgexpRsExportDest tDn="uni/fabric/path-ToSupport"/>
  <dbgexpRsTsSrc tDn="topology/pod-1/node-102/sys"/>
  <dbgexpRsTsSrc tDn="topology/pod-1/node-103/sys"/>
  <dbgexpRsTsSrc tDn="topology/pod-1/node-101/sys"/>
  <dbgexpRsData tDn="uni/fabric/tscont"/>
</dbgexpTechSupOnD>
<fabricFuncP>
  <fabricCtrlrPGrp name="default">
    <fabricRsApplTechSupOnDemand tnDbgexpTechSupOnDName=" Tech_Support_9-20-16"/>
  </fabricCtrlrPGrp>
</fabricFuncP>

```

Troubleshooting Using Atomic Counters

Atomic Counters

Atomic Counters are useful for troubleshooting connectivity between endpoints, EPGs, or an application within the fabric. A user reporting application may be experiencing slowness, or atomic counters may be needed for monitoring any traffic loss between two endpoints. One capability provided by atomic counters is the ability to place a trouble ticket into a proactive monitoring mode, for example when the problem is intermittent, and not necessarily happening at the time the operator is actively working the ticket.

Atomic counters can help detect packet loss in the fabric and allow the quick isolation of the source of connectivity issues. Atomic counters require NTP to be enabled on the fabric.

Leaf-to-leaf (TEP to TEP) atomic counters can provide the following:

- Counts of drops, admits, and excess packets
- Short-term data collection such as the last 30 seconds, and long-term data collection such as 5 minutes, 15 minutes, or more
- A breakdown of per-spine traffic (available when the number of TEPs, leaf or VPC, is less than 64)
- Ongoing monitoring

Leaf-to-leaf (TEP to TEP) atomic counters are cumulative and cannot be cleared. However, because 30 second atomic counters reset at 30 second intervals, they can be used to isolate intermittent or recurring problems.

Tenant atomic counters can provide the following:

- Application-specific counters for traffic across the fabric, including drops, admits, and excess packets
- Modes include the following:

- Endpoint to endpoint MAC address, or endpoint to endpoint IP address. Note that a single target endpoint could have multiple IP addresses associated with it.
- EPG to EPG with optional drill down
- EPG to endpoint
- EPG to * (any)
- Endpoint to external IP address



Note Atomic counters track the amount packets of between the two endpoints and use this as a measurement. They do not take into account drops or error counters in a hardware level.

Dropped packets are calculated when there are less packets received by the destination than transmitted by the source.

Excess packets are calculated when there are more packets received by the destination than transmitted by the source.

Enabling Atomic Counters

To enable using atomic counters to detect drops and misrouting in the fabric and enable quick debugging and isolation of application connectivity issues, create one or more tenant atomic counter policies, which can be one of the following types:

- EP_to_EP—Endpoint to endpoint (**dbgacEpToEp**)
- EP_to_EPG—Endpoint to endpoint group (**dbgacEpToEpg**)
- EP_to_Ext—Endpoint to external IP address (**dbgacEpToExt**)
- EPG_to_EP—Endpoint group to endpoint(**dbgacEpgToEp**)
- EPG_to_EPG—Endpoint group to endpoing group (**dbgacEpgToEpg**)
- EPG_to_IP—Endpoint group to IP address (**dbgacEpgToIp**)
- Ext_to_EP—External IP address to endpoint (**dbgacExtToEp**)
- IP_to_EPG—IP address to endpoint group (**dbgacIpToEpg**)
- Any_to_EP—Any to endpoint (**dbgacAnyToEp**)
- EP_to_Any—Endpoint to any (**dbgacEpToAny**)

Step 1 To create an EP_to_EP policy using the REST API, use XML such as the following example:

Example:

```
<dbgacEpToEp name="EP_to_EP_Policy" ownerTag="" ownerKey=""
dn="uni/tn-Tenant64/acEpToEp-EP_to_EP_Policy" descr="" adminSt="enabled">
<dbgacFilter name="EP_to_EP_Filter" ownerTag="" ownerKey="" descr=""
srcPort="https" prot="tcp" dstPort="https"/>
</dbgacEpToEp>
```

Step 2 To create an EP_to_EPG policy using the REST API, use XML such as the following example:

Example:

```
<dbgacEpToEpg name="EP_to_EPG_Pol" ownerTag="" ownerKey=""
dn="uni/tn-Tenant64/epToEpg-EP_to_EPG_Pol" descr="" adminSt="enabled">
<dbgacFilter name="EP_to_EPG_Filter" ownerTag="" ownerKey="" descr=""
srcPort="http" prot="tcp" dstPort="http"/>
<dbgacRsToAbsEpg tDn="uni/tn-Tenant64/ap-VRF64_app_prof/epg-EPG64"/>
</dbgacEpToEpg>
```

About Fabric Latency

Fabric latency is a troubleshooting tool to monitor the time taken by a packet to traverse from source to destination in the fabric. It can be used to measure latency between a combination of endpoints, endpoint groups, external interfaces, and IP addresses. Latency is measured from the **Arrival** time in the ingress leaf switch to the **Departure** time in the egress leaf switch. A prerequisite for fabric latency measurement is that all the nodes shall be synchronized with uniform time. Precision Time Protocol (PTP) is used for this, due to its' sub-microsecond accuracy, compared to NTP, which has only millisecond precisions. NTP is not sufficient to measure packet flight times within an ACI fabric, which is in the order of microseconds. Hence, latency feature requires all the nodes in the fabric to be synchronized using PTP.

There are two types of latency measurement:

- Ongoing TEP-to-TEP latency
- On-demand Tenant latency

Ongoing latency or Leaf-to-leaf (TEP to TEP) latency is used to measure latency across Tunnel End Points in leaf switches. It provides the average and maximum latency, standard deviation, and packet count computed at the destination leaf switch. The latency data collected for the last 30 seconds as well as the cumulative latency values are provided. The TEP-to-TEP latency measurements are enabled as soon as PTP is turned on in the fabric.

Tenant latency measurements can be configured to troubleshoot issues at the level of individual applications. They can be enabled for the IP flows matching a specific Flow rule programmed in the Latency TCAM. The flow rules semantics are similar to the current Atomic counter flow rules.



Note If latency measurement is configured for a particular IP flow, then the latency measurement simultaneously being done for this flow's tunnel, will not account for latency of this flow.

The following flow rules are supported for latency measurement, in addition to atomic counters:

- Measure EP to EP latency
- Measure EP to EPG latency
- Measure EP to External IP latency
- Measure EPG to EP latency
- Measure EPG to EPG latency
- Measure EPG to External IP latency

- Measure External IP to EP latency
- Measure External IP to EPG latency
- Measure Any to EP latency
- Measure External IP to External IP latency
- Measure EP to Any latency



Note Both Atomic counters and Latency measurements can be independently enabled or disabled on the same IP flow rules.

Latency data can be measured in two modes; average and histogram. The mode can be specified independently for ongoing latency as well as for each flow rule in tenant latency policies.

Average Mode

Average mode enables the following measurements.

- Average latency for last 30 seconds
- Standard deviation for last 30 seconds
- Packet count for last 30 second
- Accumulated average latency
- Accumulated Maximum latency
- Accumulated Packet Count



Note The latency measurement in average mode may slightly differ in the low order multiples, of 0.1 microsecond, compared to an external measurement equipment.

Histogram Mode

Histogram mode enables the visualization of the distribution of packet counts across different latency intervals. There are 16 Histogram buckets, and each bucket is configured with a measurement interval. Bucket 0's measurement interval is 0 to 5 microseconds, and Bucket 1 between 5 to 10 microseconds ending in 80 microseconds for the last bucket. Each of these buckets include a 64 bit counter to measure packets whose latency fell within the bucket's configured latency interval.

The histogram charts are useful for understanding the latency trends, but may not reflect the exact packet count. For measuring the actual number of packets, atomic counters may be used.

The maximum number of TEP-to-TEP latency entries supported is 384. In EX-based TORs, we can have at most 256 flows in average mode and 64 flows in histogram mode. In FX-based TORS, we can have at most 640 flows in average mode and 320 flows in histogram mode.

About PTP

Precision Time Protocol (PTP) is a time synchronization protocol defined in IEEE 1588 for nodes distributed across a network. With PTP, it is possible to synchronize distributed clocks with an accuracy of less than 1 microsecond via Ethernet networks. PTP's accuracy comes from the hardware support for PTP in the ACI fabric spines and leafs. It allows the protocol to accurately compensate for message delays and variation across the network.

PTP is a distributed protocol that specifies how real-time PTP clocks in the system synchronize with each other. These clocks are organized into a master-slave synchronization hierarchy with the grandmaster clock, which is the clock at the top of the hierarchy, determining the reference time for the entire system. Synchronization is achieved by exchanging PTP timing messages, with the members using the timing information to adjust their clocks to the time of their master in the hierarchy. PTP operates within a logical scope called a PTP domain.

The PTP process consists of two phases: establishing the master-slave hierarchy and synchronizing the clocks. Within a PTP domain, each port of an ordinary or boundary clock follows this process to determine its state:

- Examines the contents of all received announce messages (issued by ports in the master state).
- Compares the data sets of the foreign master (in the announce message) and the local clock for priority, clock class, accuracy, and so on.
- Determines its own state as either master or slave.

After the master-slave hierarchy has been established, the clocks are synchronized as follows:

- The master sends a synchronization message to the slave and notes the time it was sent.
- The slave receives the synchronization message and notes the time that it was received. For every synchronization message, there is a follow-up message. Hence, the number of sync messages should be equal to the number of follow-up messages.
- The slave sends a delay-request message to the master and notes the time it was sent.
- The master receives the delay-request message and notes the time it was received.
- The master sends a delay-response message to the slave. The number of delay request messages should be equal to the number of delay response messages.
- The slave uses these timestamps to adjust its clock to the time of its master.

In ACI fabric, when PTP feature is globally enabled in APIC, the software automatically enables PTP on specific interfaces of all the supported spines and leafs. This auto-configuration ensures that PTP is optimally enabled on all the supported nodes. In the absence of an external grandmaster clock, one of the spine switch is chosen as the grandmaster. The master spine is given a different PTP priority as compared to the other spines and leaf switches so that they will act as PTP slaves. This way we ensure that all the leaf switches in the fabric synchronize to the PTP clock of the master spine.

If an external Grandmaster clock is connected to the spines, the spine syncs to the external GM and in turn acts as a master to the leaf nodes.

PTP Default Settings

The following table lists the default settings for PTP parameters.

Parameters	Default
PTP device type	Boundary clock
PTP clock type	Two-step clock
PTP domain	0
PTP priority 1 value when advertising the clock	255
PTP priority 2 value when advertising the clock	255
PTP announce interval	1 log second
PTP announce timeout	3 announce intervals
PTP delay-request interval	0 log seconds
PTP sync interval	-2 log seconds
PTP VLAN	1



Note PTP operates only in boundary clock mode. Cisco recommends deployment of a Grand Master Clock (10 MHz) upstream, with servers containing clocks requiring synchronization connected to the switch.

PTP Verification

Command	Purpose
show ptp brief	Displays the PTP status.
show ptp clock	Displays the properties of the local clock, including clock identity.
show ptp clock foreign-masters record interface ethernet <i>slot/port</i>	Displays the state of foreign masters known to the PTP process. For each foreign master, the output displays the clock identity, basic clock properties, and whether the clock is being used as a grandmaster.
show ptp corrections	Displays the last few PTP corrections.
show ptp counters [all interface Ethernet <i>slot/port</i>]	Displays the PTP packet counters for all interfaces or for a specified interface.
show ptp parent	Displays the properties of the PTP parent.

Troubleshooting Using Atomic Counters with the REST API

Step 1 To get a list of the endpoint-to-endpoint atomic counters deployed within the fabric and the associated details such as dropped packet statistics and packet counts, use the **dbgEpToEpTsIt** class in XML such as the following example:

Example:

```
https://apic-ip-address/api/node/class/dbgEpToEpRslt.xml
```

Step 2 To get a list of external IP-to-endpoint atomic counters and the associated details, use the **dbgacExtToEp** class in XML such as the following example:

Example:

```
https://apic-ip-address/api/node/class/dbgExtToEpRslt.xml
```

Configuring Latency and PTP Using the REST API

To configure the flow policy parameters, follow the same steps for configuring atomic counters in Cisco APIC Troubleshooting guide: https://www.cisco.com/c/en/us/td/docs/switches/datacenter/aci/apic/sw/1-x/troubleshooting/b_APIC_Troubleshooting/b_APIC_Troubleshooting_chapter_0110.html#id_40942.

Step 1 To enable PTP mode:

Example:

```
/api/node/mo/uni/fabric/ptpmode.xml
<latencyPtpMode state="enabled">
```

Step 2 Configure an EP to EP policy:

Example:

```
<dbgacEpToEp name="EP_to_EP_Policy" adminSt="enabled" usage="latency-stats" latencyCollect =
"histogram">
</dbgacEpToEp>
```

Step 3 To enable both atomic counter and latency (average mode), here's the XML

Example:

```
<dbgacEpToEp name="EP_to_EP_Policy" adminSt="enabled" usage="latency-stats|atomic-counter"
latencyCollect = "average">
</dbgacEpToEp>
```

Step 4 To change the collection type for **Ongoing-mode** from average to histogram.

Example:

```
<latencyOngoingMode userMode="histogram">
```

Troubleshooting Using Faults

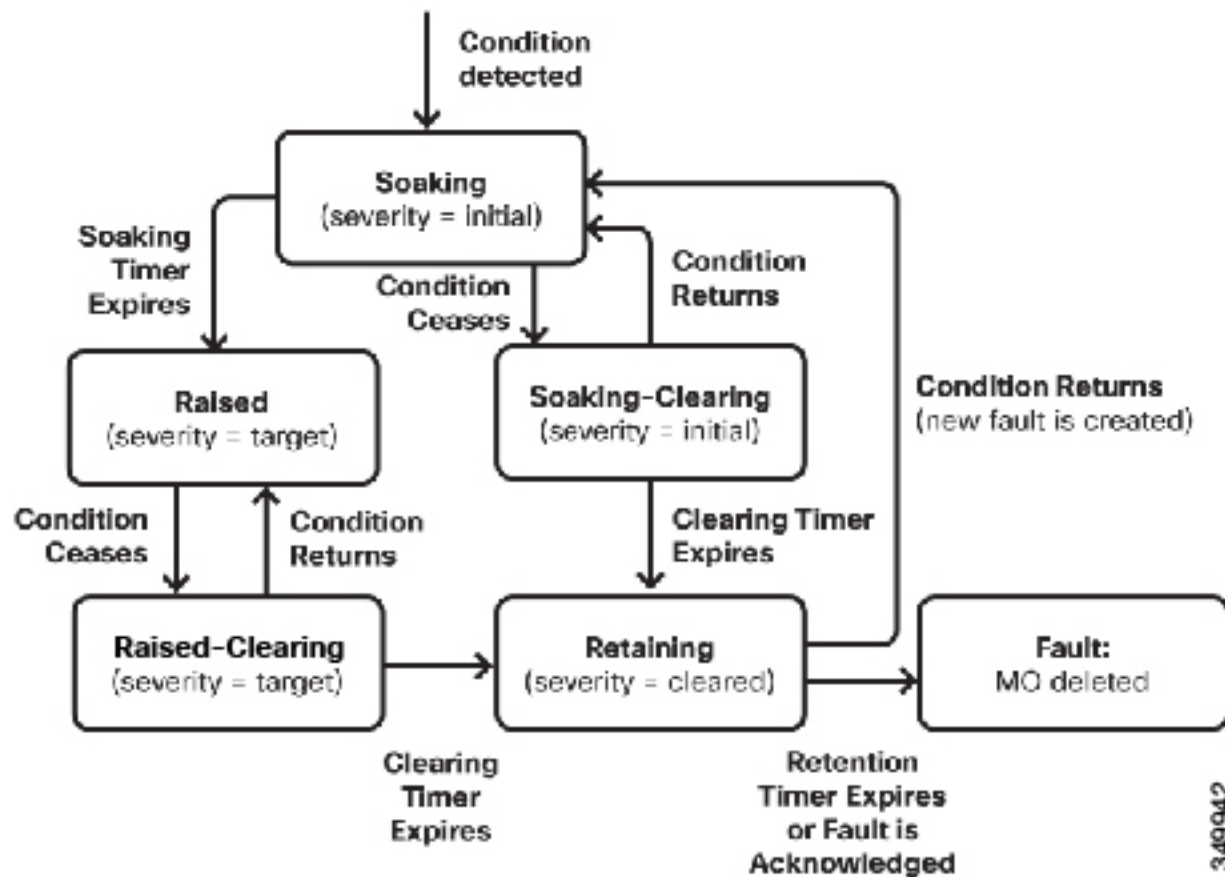
Understanding APIC Faults

From a management point of view we look at the Application Policy Infrastructure Controller (APIC) from two perspectives:

- Policy Controller - Where all fabric configuration is created, managed and applied. It maintains a comprehensive, up-to-date run-time representation of the administrative or configured state.
- Telemetry device - All devices (Fabric Switches, Virtual Switches, integrated Layer 4 to Layer 7 devices) in an Cisco Application Centric Infrastructure (ACI) fabric report faults, events and statistics to the APIC.

Faults, events, and statistics in the ACI fabric are represented as a collection of Managed Objects (MOs) within the overall ACI Object Model/Management Information Tree (MIT). All objects within ACI can be queried, including faults. In this model, a fault is represented as a mutable, stateful, and persistent MO.

Figure 5: Fault Lifecycle



When a specific condition occurs, such as a component failure or an alarm, the system creates a fault MO as a child object to the MO that is primarily associated with the fault. For a fault object class, the fault conditions are defined by the fault rules of the parent object class. Fault MOs appear as regular MOs in MIT; they have a parent, a DN, RN, and so on. The Fault "code" is an alphanumeric string in the form **FXXX**. For more information about fault codes, see the *Cisco APIC Faults, Events, and System Messages Management Guide*.

Troubleshooting Using Faults with the REST API

MOs can be queried by class and DN, with property filters, pagination, and so on.

In most cases, a fault MO is automatically created, escalated, de-escalated, and deleted by the system as specific conditions are detected. There can be at most one fault with a given code under an MO. If the same condition is detected multiple times while the corresponding fault MO is active, no additional instances of the fault MO are created. For example, if the **same condition** is detected multiple times for the **same affected object**, only **one fault** is raised while a counter for the recurrence of that fault will be incremented.

A fault MO remains in the system **until the fault condition is cleared**. For a fault to be removed, the condition raising the fault must be cleared, whether by configuration or a change in the run time state of the fabric. An exception to this is if the fault is in the cleared or retained state, in which case the fault can be deleted by the user by acknowledging it.

Severity provides an indication of the estimated impact of the condition on the capability of the system or component to provide service.

Possible values are:

- Warning (possibly no impact)
- Minor
- Major
- Critical (system or component completely unusable)

The creation of a fault MO can be triggered by internal processes such as:

- Finite state machine (FSM) transitions or detected component failures
- Conditions specified by various fault policies, some of which are user-configurable



Note You can set fault thresholds on statistical measurements such as health scores, data traffic, or temperatures.

Step 1 To retrieve the health score for a tenant named "3tierapp", send a REST query to the fabric such as the following:

Example:

```
https://apic-ip-address/api/node/mo/uni/tn-3tierapp.xml?query-target=self&rsp-subtreeinclude=health
```

Step 2 To retrieve statistics for a tenant named "3tierapp", send a REST query to the fabric such as the following:

Example:

```
https://apic-ip-address/api/node/mo/uni/tn-3tierapp.xml?query-target=self&rsp-subtreeinclude=stats
```

Step 3 To retrieve the faults for a leaf node, send a REST query to the fabric such as the following:

Example:

```
https://apic-ip-address/api/node/mo/topology/pod-1/node-103.xml?query-target=self&rsp-subtreeinclude=faults
```

Statistics

Configuring a Stats Monitoring Policy Using the REST API

To use statistics for monitoring and troubleshooting the fabric, you can configure a stats collection policy and a stats export policy to monitor many objects in the APIC.

Step 1 To create a stats collection policy using the REST API, send a POST request with XML such as the following:

Example:

```
<monEPGPol name="MonPoll" dn="uni/tn-tenant64/monepg-MonPoll">
  <monEPGTarget name="" descr="" scope="eventSevAsnP"/>
  <monEPGTarget name="" descr="" scope="faultSevAsnP"/>
  <monEPGTarget name="" descr="" scope="fvBD">
    <statsHierColl name="" descr="" histRet="inherited" granularity="5min" adminState="inherited"/>
  </monEPGTarget>
  <monEPGTarget name="" descr="" scope="syslogRsDestGroup"/>
  <monEPGTarget name="" descr="" scope="syslogSrc"/>
  <monEPGTarget name="" descr="" scope="fvCtx"/>
  <statsHierColl name="" descr="" histRet="none" granularity="1w" adminState="enabled"/>
  <statsHierColl name="" descr="" histRet="none" granularity="1qtr" adminState="enabled"/>
  <statsHierColl name="" descr="" histRet="1w" granularity="1h" adminState="enabled"/>
  <statsHierColl name="" descr="" histRet="1d" granularity="15min" adminState="enabled"/>
  <statsHierColl name="" descr="" histRet="none" granularity="1year" adminState="enabled"/>
  <statsHierColl name="" descr="" histRet="none" granularity="1mo" adminState="enabled"/>
  <statsHierColl name="" descr="" histRet="1h" granularity="5min" adminState="enabled"/>
  <statsHierColl name="" descr="" histRet="10d" granularity="1d" adminState="enabled"/>
  <syslogSrc name="VRF64_SyslogSource" descr="" minSev="warnings" incl="faults">
    <syslogRsDestGroup tDn="uni/fabric/slgrouptenant64_SyslogDest"/>
  </syslogSrc>
</monEPGPol>
```

Step 2 To configure a stats export policy send a post with XML such as the following (you can use either JSON or XML format):

Example:

```
<statsExportP
  name="" descr="" frequency="stream" format="xml" compression="gzip">
  <statsDestP name="tenant64_statsExportDest" descr="" userName="" remotePort="0"
    remotePath="192.168.100.20" protocol="sftp" host="192.168.100.1">
    <fileRsARemoteHostToEpg tDn="uni/tn-mgmt/mgmt-default/oob-default"/>
  </statsDestP>
</statsExportP>
```

```
</statsDestP>
</statsExportP>
```

Recovering a Disconnected Leaf

Recovering a Disconnected Leaf

If all fabric interfaces on a leaf are disabled (interfaces connecting a leaf to the spine) due to a configuration pushed to the leaf, connectivity to the leaf is lost forever and the leaf becomes inactive in the fabric. Trying to push a configuration to the leaf does not work because connectivity has been lost. This chapter describes how to recover a disconnected leaf.

Recovering a Disconnected Leaf Using the REST API

To recover a disconnected leaf switch, you must enable at least one of the fabric interfaces using this procedure. You can enable the remaining interfaces using the GUI, REST API, or CLI.

To enable the first interface, post a policy using the REST API to delete the policy posted and bring the fabric ports Out-of-Service. You can post a policy to the leaf switch to bring the port that is Out-of-Service to In-Service as follows:



Note This procedure assumes that 1/49 is one of the leaf switch ports connecting to the spine switch.

Step 1 Clear the block list policy from the Cisco APIC using the REST API.

Example:

```
$APIC_Address/api/policymgr/mo/.xml
<polUni>
  <fabricInst>
    <fabricOOServicePol>
      <fabricRsOosPath tDn="topology/pod-1/paths-$LEAF_Id/pathep-[eth1/49]" lc="blacklist"
status ="deleted"/>
    </fabricOOServicePol>
  </fabricInst>
</polUni>
```

Step 2 Post a local task to the node itself to bring up the interfaces you want using **l1EthIfSetInServiceLTask**.

Example:

```
$LEAF_Address/api/node/mo/topology/pod-1/node-$LEAF_Id/sys/action.xml
<actionLSubj oDn="sys/phys-[eth1/49]">
  <l1EthIfSetInServiceLTask adminSt='start'/>
</actionLSubj>
```

Troubleshooting Contracts and Taboo Contracts with Permit and Deny Logging

Verifying Contracts, Taboo Contracts, and Filters Using the REST API

This topic provides the REST API XML to verify contracts, taboo contracts, and filters.

Step 1 Verify a contract for an EPG or an external network with XML such as the following example for a provider:

Example:

```
QUERY https://apic-ip-address/api/node/class/fvRsProv.xml
```

Step 2 Verify a contract on an EPG with XML such as the following example for a consumer:

Example:

```
QUERY https://apic-ip-address/api/node/class/fvRsCons.xml
```

Step 3 Verify exported contracts using XML such as the following example:

Example:

```
QUERY https://apic-ip-address/api/node/class/vzCPif.xml
```

Step 4 Verify contracts for a VRF with XML such as the following example:

Example:

```
QUERY https://apic-ip-address/api/node/class/vzBrCP.xml
```

Step 5 Verify taboo contracts with XML such as the following example:

Example:

```
QUERY https://apic-ip-address/api/node/class/vzTaboo.xml
```

For taboo contracts for an EPG, use the same query as for contracts for EPGs.

Step 6 Verify filters using XML such as the following example:

Example:

```
QUERY https://apic-ip-address/api/node/class/vzFilter.xml
```

Viewing ACL Permit and Deny Logs Using the REST API

The following example shows how to view Layer 2 deny log data for traffic flows, using the REST API. You can send queries using the following MOs:

- acllogDropL2Flow
- acllogPermitL2Flow
- acllogDropL3Flow

- aclogPermitL3Flow
- aclogDropL2Pkt
- aclogPermitL2Pkt
- aclogDropL3Pkt
- aclogPermitL3Pkt

Before you begin

You must enable permit or deny logging, before you can view ACL contract permit and deny log data.

To view Layer 3 drop log data, send the following query using the REST API:

```
GET https://apic-ip-address/api/class/aclogDropL3Flow
```

Example:

The following example shows sample output:

```
<?xml version="1.0" encoding="UTF-8"?>
<imdata totalCount="2">
  <aclogPermitL3Flow childAction="" dn="topology/pod-1/node-101/ndbgs/aclog/tn-common/ctx-inb
/permitl3flow-spctag-333-dpctag-444-sepname-unknown-depname-unknown-sip-[100:c000:a00:700:b00:0:f00:0]
-dip-[19.0.2.10]-proto-udp-sport-17459-dport-8721-smac-00:00:15:00:00:28-dmac-00:00:12:00:00:25-sintf-
[port-channel5]-vrfencap-VXLAN: 2097153" dstEpgName="unknown" dstIp="19.0.2.10"
dstMacAddr="00:00:12:00:00:25"
dstPcTag="444" dstPort="8721" lcOwn="local" modTs="never" monPolDn="" protocol="udp"
srcEpgName="unknown"
srcIntf="port-channel5" srcIp="100:c000:a00:700:b00:0:f00:0" srcMacAddr="00:00:15:00:00:28"
srcPcTag="333"
srcPort="17459" status="" vrfEncap="VXLAN: 2097153"/>
  <aclogPermitL3Flow childAction="" dn="topology/pod-1/node-102/ndbgs/aclog/tn-common/ctx-inb
/permitl3flow-spctag-333-dpctag-444-sepname-unknown-depname-unknown-sip-[100:c000:a00:700:b00:0:f00:0]-dip-
[19.0.2.10]-proto-udp-sport-17459-dport-8721-smac-00:00:15:00:00:28-dmac-00:00:12:00:00:25-sintf-
[port-channel5]-vrfencap-VXLAN: 2097153" dstEpgName="unknown" dstIp="19.0.2.10"
dstMacAddr="00:00:12:00:00:25"
dstPcTag="444" dstPort="8721" lcOwn="local" modTs="never" monPolDn="" protocol="udp"
srcEpgName="unknown"
srcIntf="port-channel5" srcIp="100:c000:a00:700:b00:0:f00:0" srcMacAddr="00:00:15:00:00:28"
srcPcTag="333"
srcPort="17459" status="" vrfEncap="VXLAN: 2097153"/>
</imdata>
```


Troubleshooting Using Digital Optical Monitoring Statistics

Troubleshooting Using Digital Optical Monitoring With the REST API

To view DOM statistics using an XML REST API query:

Before you begin

You must have previously enabled digital optical monitoring (DOM) on an interface, before you can view the DOM statistics for it.

The following example shows how to view DOM statistics on a physical interface, eth1/25 on node-104, using a REST API query:

```
GET https://apic-ip-address/api/node/mo/topology/pod-1/node-104/sys/phys-[eth1/25]/phys/domstats.xml?
query-target=children&target-subtree-class=ethpmDOMRxPwrStats&subscription=yes
```

The following response is returned:

```
response : {
  "totalCount": "1",
  "subscriptionId": "72057611234705430",
  "imdata": [
    {"ethpmDOMRxPwrStats": {
      "attributes": {
        "alert": "none",
        "childAction": "",
        "dn": "topology/pod-1/node-104/sys/phys[eth1/25]/phys/domstats/rxpower",
        "hiAlarm": "0.158490",
        "hiWarn": "0.079430",
        "loAlarm": "0.001050",
        "loWarn": "0.002630",
        "modTs": "never",
        "status": "",
        "value": "0.139170" }}}}
```

Troubleshooting Using Port Tracking

Port Tracking Policy for Fabric Port Failure Detection

Fabric port failure detection can be enabled in the port tracking system settings. The port tracking policy monitors the status of fabric ports between leaf switches and spine switches, and ports between tier-1 leaf switches and tier-2 leaf switches. When an enabled port tracking policy is triggered, the leaf switches take down all access interfaces on the switch that have EPGs deployed on them.

If you enabled the **Include APIC ports when port tracking is triggered** option, port tracking disables Cisco Application Policy Infrastructure Controller (APIC) ports when the leaf switch loses connectivity to all fabric ports (that is, there are 0 fabric ports). Enable this feature only if the Cisco APICs are dual- or multihomed

to the fabric. Bringing down the Cisco APIC ports helps in switching over to the secondary port in the case of a dual-homed Cisco APIC.



Note Port tracking is located under **System > System Settings > Port Tracking**.

The port tracking policy specifies the number of fabric port connections that trigger the policy, and a delay timer for bringing the leaf switch access ports back up after the number of specified fabric ports is exceeded.

The following example illustrates how a port tracking policy behaves:

- The port tracking policy specifies that the threshold of active fabric port connections each leaf switch that triggers the policy is 2.
- The port tracking policy triggers when the number of active fabric port connections from the leaf switch to the spine switches drops to 2.
- Each leaf switch monitors its fabric port connections and triggers the port tracking policy according to the threshold specified in the policy.
- When the fabric port connections come back up, the leaf switch waits for the delay timer to expire before bringing its access ports back up. This gives the fabric time to reconverge before allowing traffic to resume on leaf switch access ports. Large fabrics may need the delay timer to be set for a longer time.



Note Use caution when configuring this policy. If the port tracking setting for the number of active spine ports that triggers port tracking is too high, all leaf switch access ports will be brought down.

Port Tracking Using the REST API

Before you begin

This procedure explains how to use the Port Tracking feature using the REST API.

Step 1 Turn on the Port Tracking feature using the REST API as follows (**admin state: on**):

```
<polUni>
<infraInfra dn="uni/infra">
<infraPortTrackPol name="default" delay="5" minlinks="4" adminSt="on">

</infraPortTrackPol>
</infraInfra>
</polUni>
```

Step 2 Turn off the Port Tracking feature using the REST API as follows (**admin state: off**):

```
<polUni>
<infraInfra dn="uni/infra">
<infraPortTrackPol name="default" delay="5" minlinks="4" adminSt="off">

</infraPortTrackPol>
```

```
</infraInfra>  
</polUni>
```

Removing Unwanted `_ui_` Objects

Removing Unwanted `_ui_` Objects Using the REST API

If you make changes with the Cisco NX-OS-Style CLI before using the Cisco APIC GUI, and objects appear in the Cisco APIC GUI (with names prepended with `_ui_`), these objects can be removed by performing a REST API request to the API, containing the following:

- The Class name, for example **infraAccPortGrp**
- The Dn attribute, for example **dn="uni/infra/funcprof/accportgrp-__ui_l101_eth1--31"**
- The Status attribute set to **status="deleted"**

Perform the POST to the API with the following steps:

Step 1 Log on to a user account with write access to the object to be removed.

Step 2 Send a POST to the API such as the following example:

```
POST https://192.168.20.123/api/mo/uni.xml  
Payload:<infraAccPortGrp dn="uni/infra/funcprof/accportgrp-__ui_l101_eth1--31" status="deleted"/>
```

Troubleshooting Using Contract Permit and Deny Logs

About ACL Contract Permit and Deny Logs

To log and/or monitor the traffic flow for a contract rule, you can enable and view the logging of packets or flows that were allowed to be sent because of contract permit rules or the logging of packets or flows that were dropped because of:

- Taboo contract deny rules
- Deny actions in contract subjects
- Contract or subject exceptions
- ACL contract permit in the ACI fabric is only supported on Nexus 9000 Series switches with names that end in EX or FX, and all later models. For example, N9K-C93180LC-EX or N9K-C9336C-FX.
- Deny logging in the ACI fabric is supported on all platforms.

- Using log directive on filters in management contracts is not supported. Setting the log directive will cause zoning-rule deployment failure.

For information on standard and taboo contracts and subjects, see *Cisco Application Centric Infrastructure Fundamentals* and *Cisco APIC Basic Configuration Guide*.

EPG Data Included in ACL Permit and Deny Log Output

Up to Cisco APIC, Release 3.2(1), the ACL permit and deny logs did not identify the EPGs associated with the contracts being logged. In release 3.2(1) the source EPG and destination EPG are added to the output of ACL permit and deny logs. ACL permit and deny logs include the relevant EPGs with the following limitations:

- Depending on the position of the EPG in the network, EPG data may not be available for the logs.
- When configuration changes occur, log data may be out of date. In steady state, log data is accurate.

The most accurate EPG data in the permit and deny logs results when the logs are focussed on:

- Flows from EPG to EPG, where the ingress policy is installed at the ingress TOR and the egress policy is installed at the egress TOR.
- Flows from EPG to L3Out, where one policy is applied on the border leaf TOR and the other policy is applied on a non-BL TOR.

EPGs in the log output are not supported for uSeg EPGs or for EPGs used in shared services (including shared L3Outs).

Enabling ACL Contract Permit Logging Using the REST API

The following example shows you how to enable permit and deny logging using the REST API. This example configures ACL permit and deny logging for a contract with subjects that have Permit and Deny actions configured.

For this configuration, send a post with XML similar to the following example:

Example:

```
<vzBrCP dn="uni/tn-Tenant64/brC-C64" name="C64" scope="context">
  <vzSubj consMatchT="AtleastOne" name="HTTPSsubj" provMatchT="AtleastOne" revFltPorts="yes"
rn="subj-HTTPSSbj">
    <vzRsSubjFiltAtt action="permit" directives="log" forceResolve="yes" priorityOverride="default"
rn="rssubjFiltAtt-PerHTTPS" tDn="uni/tn-Tenant64/flt-PerHTTPS" tRn="flt-PerHTTPS"
tnVzFilterName="PerHTTPS"/>
  </vzSubj>
  <vzSubj consMatchT="AtleastOne" name="httpSbj" provMatchT="AtleastOne" revFltPorts="yes"
rn="subj-httpSbj">
    <vzRsSubjFiltAtt action="deny" directives="log" forceResolve="yes" priorityOverride="default"
rn="rssubjFiltAtt-httpFilter" tDn="uni/tn-Tenant64/flt-httpFilter" tRn="flt-httpFilter"
tnVzFilterName="httpFilter"/>
  </vzSubj>
  <vzSubj consMatchT="AtleastOne" name="subj64" provMatchT="AtleastOne" revFltPorts="yes"
rn="subj-subj64">
    <vzRsSubjFiltAtt action="permit" directives="log" forceResolve="yes" priorityOverride="default"
rn="rssubjFiltAtt-icmp" tDn="uni/tn-common/flt-icmp" tRn="flt-icmp" tnVzFilterName="icmp"/>
```

```
</vzSubj>
</vzBrCP>
```

Enabling Taboo Contract Deny Logging Using the REST API

The following example shows you how to enable Taboo Contract deny logging using the REST API.

To configure taboo contract deny logging, send a post with XML similar to the following example.

Example:

```
<vzTaboo dn="uni/tn-Tenant64/taboo-TCtrctPrefix" name="TCtrctPrefix" scope="context">
  <vzTSubj name="PrefSubj" rn="tsubj-PrefSubj">
    <vzRsDenyRule directives="log" forceResolve="yes" rn="rsdenyRule-default" tCl="vzFilter"
    tDn="uni/tn-common/flt-default" tRn="flt-default"/>
  </vzTSubj>
</vzTaboo>
```

Viewing ACL Permit and Deny Logs Using the REST API

The following example shows how to view Layer 2 deny log data for traffic flows, using the REST API. You can send queries using the following MOs:

- aclogDropL2Flow
- aclogPermitL2Flow
- aclogDropL3Flow
- aclogPermitL3Flow
- aclogDropL2Pkt
- aclogPermitL2Pkt
- aclogDropL3Pkt
- aclogPermitL3Pkt

Before you begin

You must enable permit or deny logging, before you can view ACL contract permit and deny log data.

To view Layer 3 drop log data, send the following query using the REST API:

```
GET https://apic-ip-address/api/class/aclogDropL3Flow
```

Example:

The following example shows sample output:

```
<?xml version="1.0" encoding="UTF-8"?>
<imdata totalCount="2">
```

```

    <acllogPermitL3Flow childAction="" dn="topology/pod-1/node-101/ndbgs/acllog/tn-common/ctx-inb
/permitl3flow-spctag-333-dpctag-444-sepgname-unknown-depgname-unknown-sip-[100:c000:a00:700:b00:0:f00:0]
-dip-[19.0.2.10]-proto-udp-sport-17459-dport-8721-smac-00:00:15:00:00:28-dmac-00:00:12:00:00:25-sintf-
    [port-channel5]-vrfencap-VXLAN: 2097153" dstEpgName="unknown" dstIp="19.0.2.10"
dstMacAddr="00:00:12:00:00:25"
    dstPcTag="444" dstPort="8721" lcOwn="local" modTs="never" monPolDn="" protocol="udp"
srcEpgName="unknown"
    srcIntf="port-channel5" srcIp="100:c000:a00:700:b00:0:f00:0" srcMacAddr="00:00:15:00:00:28"
srcPcTag="333"
    srcPort="17459" status="" vrfEncap="VXLAN: 2097153"/>
    <acllogPermitL3Flow childAction="" dn="topology/pod-1/node-102/ndbgs/acllog/tn-common/ctx-inb
/permitl3flow-spctag-333-dpctag-444-sepgname-unknown-depgname-unknown-sip-[100:c000:a00:700:b00:0:f00:0]-dip-
[19.0.2.10]-proto-udp-sport-17459-dport-8721-smac-00:00:15:00:00:28-dmac-00:00:12:00:00:25-sintf-
    [port-channel5]-vrfencap-VXLAN: 2097153" dstEpgName="unknown" dstIp="19.0.2.10"
dstMacAddr="00:00:12:00:00:25"
    dstPcTag="444" dstPort="8721" lcOwn="local" modTs="never" monPolDn="" protocol="udp"
srcEpgName="unknown"
    srcIntf="port-channel5" srcIp="100:c000:a00:700:b00:0:f00:0" srcMacAddr="00:00:15:00:00:28"
srcPcTag="333"
    srcPort="17459" status="" vrfEncap="VXLAN: 2097153"/>
</imdata>

```



PART III

Part 3: Setting Up APIC and the Fabric Using the REST API

- [Managing APIC Clusters, on page 143](#)
- [Managing Fabrics, on page 149](#)
- [Configuring Tenant Policies, on page 153](#)
- [Provisioning Core Services, on page 201](#)
- [Provisioning Layer 2 Networks, on page 225](#)
- [Provisioning Layer 3 Outside Connections, on page 305](#)
- [Configuring QoS, on page 405](#)
- [Managing Layer 4 to Layer 7 Services, on page 413](#)
- [Configuring Security, on page 457](#)
- [Creating Quota Management, on page 491](#)
- [Configuring a Forwarding Scale Profile Policy, on page 493](#)



CHAPTER 10

Managing APIC Clusters

- [Cluster Management Guidelines](#), on page 143
- [Expanding and Contracting Clusters](#), on page 144
- [Managing Cluster High Availability](#), on page 146

Cluster Management Guidelines

Cluster Management Guidelines

The Cisco Application Policy Infrastructure Controller (APIC) cluster is comprised of multiple Cisco APICs that provide operators a unified real time monitoring, diagnostic, and configuration management capability for the Cisco Application Centric Infrastructure (ACI) fabric. To assure optimal system performance, follow the guidelines below for making changes to the Cisco APIC cluster.



Note Prior to initiating a change to the cluster, always verify its health. When performing planned changes to the cluster, all controllers in the cluster should be healthy. If one or more of the Cisco APICs' health status in the cluster is not "fully fit", remedy that situation before proceeding. Also, assure that cluster controllers added to the Cisco APIC are running the same version of firmware as the other controllers in the Cisco APIC cluster.

Follow these general guidelines when managing clusters:

- We recommend that you have at least 3 active Cisco APICs in a cluster, along with additional standby Cisco APICs. Cisco APIC clusters can have from 3 to 7 active Cisco APICs. Refer to the [Verified Scalability Guide](#) to determine how many active Cisco APICs are required for your deployment.
- Disregard cluster information from Cisco APICs that are not currently in the cluster; they do not provide accurate cluster information.
- Cluster slots contain a Cisco APIC `ChassisID`. Once you configure a slot, it remains unavailable until you decommission the Cisco APIC with the assigned `ChassisID`.
- If a Cisco APIC firmware upgrade is in progress, wait for it to complete and the cluster to be fully fit before proceeding with any other changes to the cluster.
- When moving a Cisco APIC, first ensure that you have a healthy cluster. After verifying the health of the Cisco APIC cluster, choose the Cisco APIC you intend to shut down. After the Cisco APIC has

shutdown, move the Cisco APIC, re-connect it, and then turn it back on. From the GUI, verify that the all controllers in the cluster return to a fully fit state.



Note Only move one Cisco APIC at a time.

- When moving a Cisco APIC that is connected to a set of leaf switches to another set of leaf switches or when moving a Cisco APIC to different port within the same leaf switch, first ensure that you have a healthy cluster. After verifying the health of the Cisco APIC cluster, choose the Cisco APIC that you intend to move and decommission it from the cluster. After the Cisco APIC is decommissioned, move the Cisco APIC and then commission it.
- Before configuring the Cisco APIC cluster, ensure that all the Cisco APICs are running the same firmware version. Initial clustering of Cisco APICs running differing versions is an unsupported operation and may cause problems within the cluster.
- Unlike other objects, log record objects are stored only in one shard of a database on one of the Cisco APICs. These objects get lost forever if you decommission or replace that Cisco APIC.
- When you decommission a Cisco APIC, the Cisco APIC loses all fault, event, and audit log history that was stored in it. If you replace all Cisco APICs, you lose all log history. Before you migrate a Cisco APIC, we recommend that you manually backup the log history.

Expanding and Contracting Clusters

Expanding the APIC Cluster Size

Follow these guidelines to expand the APIC cluster size:

- Schedule the cluster expansion at a time when the demands of the fabric workload will not be impacted by the cluster expansion.
- If one or more of the APIC controllers' health status in the cluster is not "fully fit", remedy that situation before proceeding.
- Stage the new APIC controller(s) according to the instructions in their hardware installation guide. Verify in-band connectivity with a PING test.
- Increase the cluster target size to be equal to the existing cluster size controller count plus the new controller count. For example, if the existing cluster size controller count is 3 and you are adding 3 controllers, set the new cluster target size to 6. The cluster proceeds to sequentially increase its size one controller at a time until all new the controllers are included in the cluster.



Note Cluster expansion stops if an existing APIC controller becomes unavailable. Resolve this issue before attempting to proceed with the cluster expansion.

- Depending on the amount of data the APIC must synchronize upon the addition of each appliance, the time required to complete the expansion could be more than 10 minutes per appliance. Upon successful expansion of the cluster, the APIC operational size and the target size will be equal.



Note Allow the APIC to complete the cluster expansion before making additional changes to the cluster.

Expanding the Cisco APIC Cluster

Expanding the Cisco APIC cluster is the operation to increase any size mismatches, from a cluster size of N to size N+1, within legal boundaries. The operator sets the administrative cluster size and connects the APICs with the appropriate cluster IDs, and the cluster performs the expansion.

During cluster expansion, regardless of in which order you physically connect the APIC controllers, the discovery and expansion takes place sequentially based on the APIC ID numbers. For example, APIC2 is discovered after APIC1, and APIC3 is discovered after APIC2 and so on until you add all the desired APICs to the cluster. As each sequential APIC is discovered, a single data path or multiple data paths are established, and all the switches along the path join the fabric. The expansion process continues until the operational cluster size reaches the equivalent of the administrative cluster size.

Expanding the APIC Cluster Using the REST API

The cluster drives its actual size to the target size. If the target size is higher than the actual size, the cluster size expands.

Step 1 Set the target cluster size to expand the APIC cluster size.

Example:

```
POST
https://<IP address>/api/node/mo/uni/controller.xml
<infraClusterPol name='default' size=3/>
```

Step 2 Physically connect the APIC controllers that you want to add to the cluster.

Contracting the Cisco APIC Cluster

Contracting the Cisco APIC cluster is the operation to decrease any size mismatches, from a cluster size of N to size N-1, within legal boundaries. As the contraction results in increased computational and memory load for the remaining APICs in the cluster, the decommissioned APIC cluster slot becomes unavailable by operator input only.

During cluster contraction, you must begin decommissioning the last APIC in the cluster first and work your way sequentially in reverse order. For example, APIC4 must be decommissioned before APIC3, and APIC3 must be decommissioned before APIC2.

Contracting the APIC Cluster Using the REST API

The cluster drives its actual size to the target size. If the target size is lower than the actual size, the cluster size contracts.

Step 1 Set the target cluster size so as to contract the APIC cluster size.

Example:

```
POST
https://<IP address>/api/node/mo/uni/controller.xml
<infraClusterPol name='default' size=1/>
```

Step 2 Decommission APIC3 on APIC1 for cluster contraction.

Example:

```
POST
https://<IP address>/api/node/mo/topology/pod-1/node-1/av.xml
<infraWiNode id=3 adminSt='out-of-service'/>
```

Step 3 Decommission APIC2 on APIC1 for cluster contraction.

Example:

```
POST
https://<IP address>/api/node/mo/topology/pod-1/node-1/av.xml
<infraWiNode id=2 adminSt='out-of-service'/>
```

Managing Cluster High Availability

About Cold Standby for a Cisco APIC Cluster

The Cold Standby functionality for a Cisco Application Policy Infrastructure Controller (APIC) cluster enables you to operate the Cisco APICs in a cluster in an Active/Standby mode. In a Cisco APIC cluster, the designated active Cisco APICs share the load and the designated standby Cisco APICs can act as a replacement for any of the Cisco APICs in the active cluster.

As an admin user, you can set up the Cold Standby functionality when the Cisco APIC is launched for the first time. We recommend that you have at least three active Cisco APICs in a cluster, and one or more standby Cisco APICs. As an admin user, you can initiate the switch over to replace an active Cisco APIC with a standby Cisco APIC.

Important Notes

- The standby Cisco APICs are automatically updated with firmware updates to keep the backup Cisco APIC at same firmware version as the active cluster.
- During an upgrade process, after all the active Cisco APICs are upgraded, the standby Cisco APICs are also upgraded automatically.
- Temporary IDs are assigned to the standby Cisco APICs. After a standby Cisco APIC is switched over to an active Cisco APIC, a new ID is assigned.

- The admin login is not enabled on the standby Cisco APICs. To troubleshoot a Cold Standby Cisco APIC, you must log in to the standby using SSH as *rescue-user*.
- During the switch over, the replaced active Cisco APIC is powered down to prevent connectivity to the replaced Cisco APIC.
- Switch over fails under the following conditions:
 - If there is no connectivity to the standby Cisco APIC.
 - If the firmware version of the standby Cisco APIC is not the same as that of the active cluster.
- After switching over a standby Cisco APIC to be active, if it was the only standby, you must configure a new standby.
- The following limitations are observed for retaining out of band address for the standby Cisco APIC after a fail over:
 - The standby (new active) Cisco APIC may not retain its out of band address if more than 1 active Cisco APICs are down or unavailable.
 - The standby (new active) Cisco APIC may not retain its out of band address if it is in a different subnet than the active Cisco APIC. This limitation is only applicable for Cisco APIC release 2.x.
 - The standby (new active) Cisco APIC may not retain its IPv6 out of band address. This limitation is not applicable starting from Cisco APIC release 3.1x.
 - The standby (new active) Cisco APIC may not retain its out of band address if you have configured a non-static OOB management IP address policy for the replacement (old active) Cisco APIC.
 - The standby (new active) Cisco APIC may not retain its out of band address if it is not in a pod that has an active Cisco APIC.



Note If you want to retain the standby Cisco APIC's out of band address despite the limitations, you must manually change the OOB policy for the replaced Cisco APIC after the replace operation had completed successfully.

- There must be three active Cisco APICs to add a standby Cisco APIC.
- The standby Cisco APIC does not participate in policy configuration or management.
- No information is replicated to the standby Cisco APICs, not even the administrator credentials.

Switching Over Active APIC with Standby APIC Using REST API

Use this procedure to switch over an active APIC with standby APIC using REST API.

Switch over active APIC with standby APIC.

URL for POST: `https://ip_address/api/node/mo/topology/pod-initiator_pod_id/node-initiator_id/av.xml`
 Body: `<infraWiNode id=outgoing_apic_id targetMbSn=backup-serial-number/>`
 where `initiator_id` = id of an active APIC other than the APIC being replaced.

```
pod-initiator_pod_id = pod ID of the active APIC  
backup-serial-number = serial number of standby APIC
```

Example:

```
https://ip address/api/node/mo/topology/pod-1/node-1/av.xml  
<infraWNode id=2 targetMbSn=FCH1750V00Q/>
```



CHAPTER 11

Managing Fabrics

- [Maintenance Mode](#), on page 149
- [Removing a Switch to Maintenance Mode Using the REST API](#), on page 151
- [Inserting a Switch to Operation Mode Using the CLI](#), on page 151

Maintenance Mode

Following are terms that are helpful to understand when using maintenance mode:

- **Maintenance mode:** Used to isolate a switch from user traffic for debugging purposes. You can put a switch in **maintenance mode** by enabling the **Maintenance (GIR)** field in the **Fabric Membership** page in the APIC GUI, located at **Fabric > Inventory > Fabric Membership** (right-click on a switch and choose **Maintenance (GIR)**).

If you put a switch in **maintenance mode**, that switch is not considered as a part of the operational ACI fabric infra and it will not accept regular APIC communications.

You can use maintenance mode to gracefully remove a switch and isolate it from the network in order to perform debugging operations. The switch is removed from the regular forwarding path with minimal traffic disruption.

In graceful removal, all external protocols are gracefully brought down except the fabric protocol (IS-IS) and the switch is isolated from the network. During maintenance mode, the maximum metric is advertised in IS-IS within the Cisco Application Centric Infrastructure (Cisco ACI) fabric and therefore the leaf switch in maintenance mode does not attract traffic from the spine switches. In addition, all front-panel interfaces on the switch are shutdown except for the fabric interfaces. To return the switch to its fully operational (normal) mode after the debugging operations, you must recommission the switch. This operation will trigger a stateless reload of the switch.

In graceful insertion, the switch is automatically decommissioned, rebooted, and recommissioned. When recommissioning is completed, all external protocols are restored and maximum metric in IS-IS is reset after 10 minutes.

The following protocols are supported:

- Border Gateway Protocol (BGP)
- Enhanced Interior Gateway Routing Protocol (EIGRP)
- Intermediate System-to-Intermediate System (IS-IS)

- Open Shortest Path First (OSPF)
- Link Aggregation Control Protocol (LACP)

Protocol Independent Multicast (PIM) is not supported.

Important Notes

- If a border leaf switch has a static route and is placed in maintenance mode, the route from the border leaf switch might not be removed from the routing table of switches in the ACI fabric, which causes routing issues.
To work around this issue, either:
 - Configure the same static route with the same administrative distance on the other border leaf switch, or
 - Use IP SLA or BFD for track reachability to the next hop of the static route
- While the switch is in maintenance mode, the Ethernet port module stops propagating the interface related notifications. As a result, if the remote switch is rebooted or the fabric link is flapped during this time, the fabric link will not come up afterward unless the switch is manually rebooted (using the **acdiag touch clean** command), decommissioned, and recommissioned.
- While the switch is in maintenance mode, CLI 'show' commands on the switch show the front panel ports as being in the up state and the BGP protocol as up and running. The interfaces are actually shut and all other adjacencies for BGP are brought down, but the displayed active states allow for debugging.
- For multi-pod / multi-site, **IS-IS metric for redistributed routes** should be set to less than 63 to minimize the traffic disruption when bringing the node back into the fabric. To set the **IS-IS metric for redistributed routes**, choose **Fabric > Fabric Policies > Pod Policies > IS-IS Policy**.
- Existing GIR supports all Layer 3 traffic diversion. With LACP, all the Layer 2 traffic is also diverted to the redundant node. Once a node goes into maintenance mode, LACP running on the node immediately informs neighbors that it can no longer be aggregated as part of port-channel. All traffic is then diverted to the vPC peer node.
- The following operations are not allowed in maintenance mode:
 - **Upgrade**: Upgrading the network to a newer version
 - **Stateful Reload**: Restarting the GIR node or its connected peers
 - **Stateless Reload**: Restarting with a clean configuration or power-cycle of the GIR node or its connected peers
 - **Link Operations**: Shut / no-shut or optics OIR on the GIR node or its peer node
 - **Configuration Change**: Any configuration change (such as clean configuration, import, or snapshot rollback)
 - **Hardware Change**: Any hardware change (such as adding, swapping, removing FRU's or RMA)

Removing a Switch to Maintenance Mode Using the REST API

Use this procedure to remove a switch to maintenance mode using the REST API.

Remove a switch to maintenance mode.

Example:

```
POST
https://<IP address>/api/node/mo/uni/fabric/outofsvc.xml

<fabricOOServicePol
  descr=""
  dn=""
  name="default"
  nameAlias=""
  ownerKey=""
  ownerTag="">
  <fabricRsDecommissionNode
    debug="yes"
    dn=""
    removeFromController="no"
    tDn="topology/pod-1/node-102"/>
</fabricOOServicePol>
```

Inserting a Switch to Operation Mode Using the CLI

Use this procedure to insert a switch to operational mode using the CLI.

[no]no debug-switch *node_id or node_name*

Inserts the switch to operational mode.



CHAPTER 12

Configuring Tenant Policies

- [Basic Tenant Configuration](#), on page 153
- [Tenants in Multiple Private Networks](#), on page 154
- [Tenant Policy Example](#), on page 158
- [EPGs](#), on page 168
- [Intra-EPG Isolation](#), on page 171
- [Microsegmentation](#), on page 177
- [Application Profiles](#), on page 180
- [Contracts, Taboo Contracts, and Preferred Groups](#), on page 184
- [Configuring an Enforced Bridge Domain](#), on page 199

Basic Tenant Configuration

Creating a Tenant, VRF, and Bridge Domain Using the REST API

SUMMARY STEPS

1. Create a tenant.
2. Create a VRF and bridge domain.

DETAILED STEPS

Step 1 Create a tenant.

Example:

```
POST https://apic-ip-address/api/mo/uni.xml
<fvTenant name="ExampleCorp"/>
```

When the POST succeeds, you see the object that you created in the output.

Step 2 Create a VRF and bridge domain.

Note The Gateway Address can be an IPv4 or an IPv6 address. For more about details IPv6 gateway address, see the related KB article, *KB: Creating a Tenant, VRF, and Bridge Domain with IPv6 Neighbor Discovery*.

Example:

URL for POST: `https://apic-ip-address/api/mo/uni/tn-ExampleCorp.xml`

```
<fvTenant name="ExampleCorp">
  <fvCtx name="pvn1"/>
  <fvBD name="bd1">
    <fvRsCtx tnFvCtxName="pvn1"/>
    <fvSubnet ip="10.10.100.1/24"/>
  </fvBD>
</fvTenant>
```

Note If you have a public subnet when you configure the routed outside, you must associate the bridge domain with the outside configuration.

Tenants in Multiple Private Networks

About Multiple Private Networks with Inter-Tenant Communication

- This use case may be typical for environments where an ACI administrator wishes to create multiple tenants with the ability to support inter-tenant communications.

This method has the following advantages and disadvantages:

Advantages:

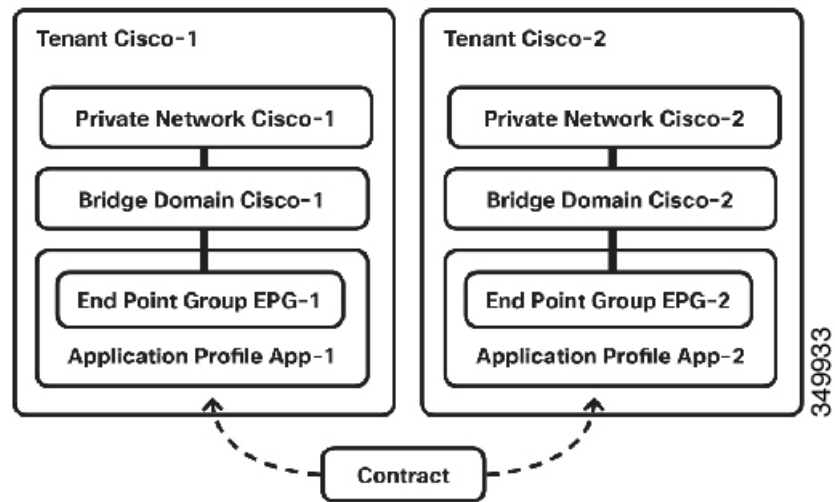
- Each tenant container can be managed separately
- Allows for maximum isolation between tenants

Disadvantages:

- Tenant address space must be unique

From a containment and relationship perspective, this topology looks as follows:

Figure 6: Multiple Private Networks with Inter-Tenant Communication



Configuring Multiple Private Networks with Inter-Tenant Communication Using the REST API

Configure the Cisco-1 and Cisco-2 private networks, with communication between them, using the REST API in the following steps:

Step 1 Configure Cisco-1 tenant using the following XML posted to the APIC REST API:

Example:

```
<fvTenant dn="uni/tn-Cisco1" name="Cisco1">
  <vzBrCP name="ICMP" scope="global">
    <vzSubj consMatchT="AtleastOne" name="icmp" provMatchT="AtleastOne"
      revFltPorts="yes">
      <vzRsSubjFiltAtt tnVzFilterName="icmp"/>
    </vzSubj>
  </vzBrCP>

  <vzCPIf dn="uni/tn-Cisco1/cif-ICMP" name="ICMP">
    <vzRsIf consMatchT="AtleastOne" name="icmp" provMatchT="AtleastOne"
      revFltPorts="yes">
      <vzRsSubjFiltAtt tDn="uni/tn-Cisco2/brc-default"/>
    </vzRsIf>
  </vzCPIf>
  <fvCtx knwMcastAct="permit" name="CiscoCtx" pcEnfPref="enforced"/>

  <fvBD arpFlood="yes" mac="00:22:BD:F8:19:FF" name="CiscoBD2" unicastRoute="yes"
    unkMacUcastAct="flood" unkMcastAct="flood">
    <fvRsCtx tnFvCtxName="CiscoCtx2"/>
  </fvBD>
</fvTenant>
```

```

<fvBD arpFlood="yes" name="CiscoBD" unicastRoute="yes" unkMacUcastAct="flood"
unkMcastAct="flood">
<fvRsCtx tnFvCtxName="CiscoCtx"/>
</fvBD>
<fvAp name="CCO">
<fvAEPg matchT="AtleastOne" name="EPG1">
<fvRsPathAtt encap="vlan-1202" instrImedcy="immediate" mode="native"
tDn="topology/pod-1/paths-202/pathep-[eth1/2]"/>
<fvSubnet ip="172.16.1.1/24" scope="private,shared"/>

<fvRsDomAtt instrImedcy="lazy" resImedcy="lazy" tDn="uni/phys-
PhysDomainforCisco"/>

<fvRsBd tnFvBDName="CiscoBD"/>
<fvRsProv matchT="AtleastOne" tnVzBrCPName="ICMP"/>
</fvAEPg>
</fvAp>
</fvTenant>

<fvRsBd tnFvBDName="CiscoBD"/>
<fvRsProv matchT="AtleastOne" tnVzBrCPName="ICMP"/>
</fvAEPg>
</fvAp>
</fvTenant>

```

Step 2 Configure Cisco-2 tenant using the following XML posted to the APIC REST API:

Example:

```

<fvTenant dn="uni/tn-Cisco2" name="Cisco2">
<fvCtx knwMcastAct="permit" name="CiscoCtx" pcEnfPref="enforced"/>
<fvBD arpFlood="yes" mac="00:22:BD:F8:19:FF" name="CiscoBD2" unicastRoute="yes"
unkMacUcastAct="flood" unkMcastAct="flood">
<fvRsCtx tnFvCtxName="CiscoCtx"/>
</fvBD>
<fvBD arpFlood="yes" name="CiscoBD" unicastRoute="yes" unkMacUcastAct="flood"
unkMcastAct="flood">
<fvRsCtx tnFvCtxName="CiscoCtx"/>
</fvBD>
<fvAp name="CCO">
<fvAEPg matchT="AtleastOne" name="EPG2">
<fvRsPathAtt encap="vlan-1202" instrImedcy="immediate" mode="native"
tDn="topology/pod-1/paths-201/pathep-[eth1/2]"/>
<fvSubnet ip="172.16.1.1/24" scope="private,shared"/>

<fvRsDomAtt instrImedcy="lazy" resImedcy="lazy" tDn="uni/phys-
PhysDomainforCisco"/>

<fvRsBd tnFvBDName="CiscoBD"/>
<fvRsConsIf matchT="AtleastOne" tnVzBrCPIfName="ICMP"/>
</fvAEPg>
</fvAp>
</fvTenant>

```

About Multiple Private Networks with Intra-Tenant Communication

Another use case that may be desirable to support is the option to have a single tenant with multiple private networks. This may be a result of needing to provide multitenancy at a network level, but not at a management

level. It may also be caused by needing to support overlapping subnets within a single tenant, due to mergers and acquisitions or other business changes.

This method has the following advantages and disadvantages:

Advantages:

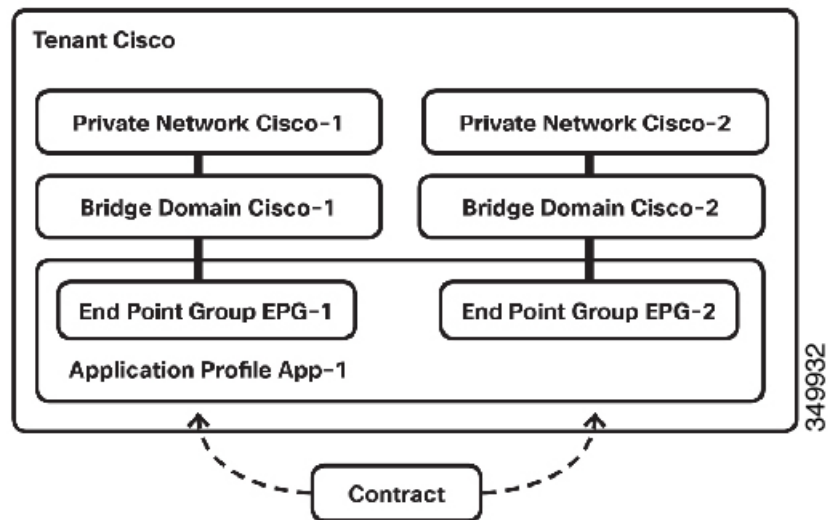
- Ability to have overlapping subnets within a single tenant

Disadvantages:

- EPGs residing in overlapping subnets cannot have policy applied between one another

The object containment for this particular setup can be depicted as shown below:

Figure 7: Multiple Private Networks with Intra-Tenant Communication



Configuring Multiple Tenants with Intra-Tenant Communication Using the REST API

SUMMARY STEPS

1. Configure the Tenant Cisco, with Cisco-1 and Cisco-2 networks, using the following XML posted to the APIC REST API:

DETAILED STEPS

Configure the Tenant Cisco, with Cisco-1 and Cisco-2 networks, using the following XML posted to the APIC REST API:

Example:

```

<fvTenant dn="uni/tn-Cisco" name="Cisco">
<vzBrCP name="ICMP" scope="tenant">
<vzSubj consMatchT="AtleastOne" name="icmp" provMatchT="AtleastOne"
revFltPorts="yes">
<vzRsSubjFiltAtt tnVzFilterName="icmp"/>
</vzSubj>
</vzBrCP>
<fvCtx knwMcastAct="permit" name="CiscoCtx" pcEnfPref="enforced"/>
<fvCtx knwMcastAct="permit" name="CiscoCtx2" pcEnfPref="enforced"/>
<fvBD arpFlood="yes" mac="00:22:BD:F8:19:FF" name="CiscoBD2" unicastRoute="yes"
unkMacUcastAct="flood" unkMcastAct="flood">
<fvRsCtx tnFvCtxName="CiscoCtx2"/>
</fvBD>
<fvBD arpFlood="yes" mac="00:22:BD:F8:19:FF" name="CiscoBD" unicastRoute="yes"
unkMacUcastAct="flood" unkMcastAct="flood">
<fvRsCtx tnFvCtxName="CiscoCtx"/>
</fvBD>
<fvAp name="CCO">
<fvAEPg matchT="AtleastOne" name="Web">
<fvRsCons tnVzBrCPName="ICMP"/>
<fvRsPathAtt encap="vlan-1201" instrImedcy="immediate" mode="native"
tDn="topology/pod-1/paths-201/pathep-[eth1/16]"/>
<fvSubnet ip="172.16.2.1/24" scope="private,shared"/>
<fvRsDomAtt instrImedcy="lazy" resImedcy="lazy" tDn="uni/phys-
PhysDomainforCisco"/>
<fvRsBd tnFvBDName="CiscoBD2"/>
</fvAEPg>
<fvAEPg matchT="AtleastOne" name="App">
<fvRsPathAtt encap="vlan-1202" instrImedcy="immediate" mode="native"
tDn="topology/pod-1/paths-202/pathep-[eth1/2]"/>
<fvSubnet ip="172.16.1.1/24" scope="private,shared"/>
<fvRsDomAtt instrImedcy="lazy" resImedcy="lazy" tDn="uni/phys-
PhysDomainforCisco"/>
<fvRsBd tnFvBDName="CiscoBD"/>
<fvRsProv matchT="AtleastOne" tnVzBrCPName="ICMP"/>
</fvAEPg>
</fvAp>
</fvTenant>

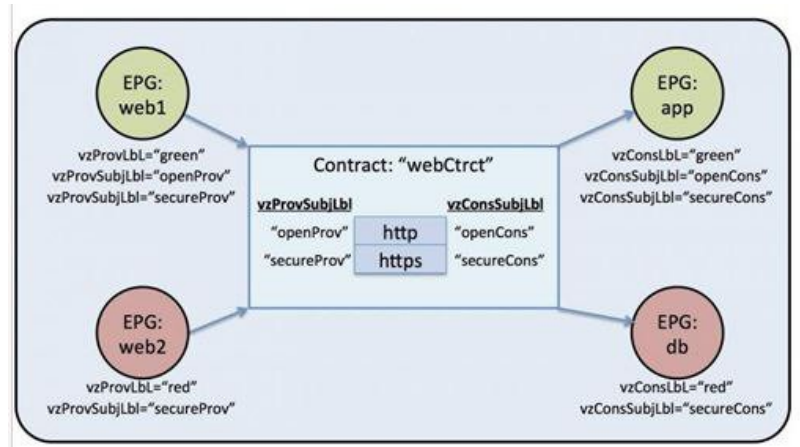
```

Tenant Policy Example

Tenant Policy Example Overview

The description of the tenant policy example in this appendix uses XML terminology (http://en.wikipedia.org/wiki/XML#Key_terminology). This example demonstrates how basic APIC policy model constructs are rendered into the XML code. The following figure provides an overview of the tenant policy example.

Figure 8: EPGs and Contract Contained in Tenant Solar



In the figure, according to the contract called webCtrct and the EPG labels, the green-labeled EPG:web1 can communicate with green-labeled EPG:app using both http and https, the red-labeled EPG:web2 can communicate with the red-labeled EPG:db using only https.

Tenant Policy Example XML Code

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

    <vzFilter name="Http">
      <vzEntry name="e1"

        etherT="ipv4"
        prot="tcp"
        dFromPort="80"
        dToPort="80"/>
    </vzFilter>

    <vzFilter name="Https">
      <vzEntry name="e1"
        etherT="ipv4"
        prot="tcp"
        dFromPort="443"
        dToPort="443"/>
    </vzFilter>

    <vzBrCP name="webCtrct">
      <vzSubj name="http" revFltPorts="true" provmatchT="All">
        <vzRsSubjFiltAtt tnVzFilterName="Http"/>
        <vzRsSubjGraphAtt graphName="G1" termNodeName="TProv"/>
        <vzProvSubjLbl name="openProv"/>
        <vzConsSubjLbl name="openCons"/>
      </vzSubj>
      <vzSubj name="https" revFltPorts="true" provmatchT="All">
        <vzProvSubjLbl name="secureProv"/>
        <vzConsSubjLbl name="secureCons"/>
        <vzRsSubjFiltAtt tnVzFilterName="Https"/>
        <vzRsOutTermGraphAtt graphName="G2" termNodeName="TProv"/>
      </vzSubj>
    </vzBrCP>
  </fvTenant>
</polUni>
```

```

<fvCtx name="solarctx1"/>

<fvBD name="solarBD1">
  <fvRsCtx tnFvCtxName="solarctx1" />
  <fvSubnet ip="11.22.22.20/24">
    <fvRsBDSubnetToProfile
      tnL3extOutName="rout1"
      tnRtctrlProfileName="profExport"/>
  </fvSubnet>
  <fvSubnet ip="11.22.22.211/24">
    <fvRsBDSubnetToProfile
      tnL3extOutName="rout1"
      tnRtctrlProfileName="profExport"/>
  </fvSubnet>
</fvBD>

<fvAp name="sap">
  <fvAEPg name="web1">
    <fvRsBd tnFvBDName="solarBD1" />
    <fvRsDomAtt tDn="uni/vmmp-VMware/dom-mininet" />
    <fvRsProv tnVzBrCPName="webCtrct" matchT="All">
      <vzProvSubjLbl name="openProv"/>
      <vzProvSubjLbl name="secureProv"/>
      <vzProvLbl name="green"/>
    </fvRsProv>
  </fvAEPg>
  <fvAEPg name="web2">
    <fvRsBd tnFvBDName="solarBD1" />
    <fvRsDomAtt tDn="uni/vmmp-VMware/dom-mininet" />
    <fvRsProv tnVzBrCPName="webCtrct" matchT="All">
      <vzProvSubjLbl name="secureProv"/>
      <vzProvLbl name="red"/>
    </fvRsProv>
  </fvAEPg>
  <fvAEPg name="app">
    <fvRsBd tnFvBDName="solarBD1" />
    <fvRsDomAtt tDn="uni/vmmp-VMware/dom-mininet" />
    <fvRsCons tnVzBrCPName="webCtrct">
      <vzConsSubjLbl name="openCons"/>
      <vzConsSubjLbl name="secureCons"/>
      <vzConsLbl name="green"/>
    </fvRsCons>
  </fvAEPg>
  <fvAEPg name="db">
    <fvRsBd tnFvBDName="solarBD1" />
    <fvRsDomAtt tDn="uni/vmmp-VMware/dom-mininet" />
    <fvRsCons tnVzBrCPName="webCtrct">
      <vzConsSubjLbl name="secureCons"/>
      <vzConsLbl name="red"/>
    </fvRsCons>
  </fvAEPg>
</fvAp>
</fvTenant>
</polUni>

```

Tenant Policy Example Explanation

This section contains a detailed explanation of the tenant policy example.

Policy Universe

The policy universe contains all the tenant-managed objects where the policy for each tenant is defined.

```
<polUni>
```

This starting tag, `<polUni>`, in the first line indicates the beginning of the policy universe element. This tag is matched with `</polUni>` at the end of the policy. Everything in between is the policy definition.

Tenant Policy Example

The `<fvTenant>` tag identifies the beginning of the tenant element.

```
<fvTenant name="solar">
```

All of the policies for this tenant are defined in this element. The name of the tenant in this example is solar. The tenant name must be unique in the system. The primary elements that the tenant contains are filters, contracts, outside networks, bridge domains, and application profiles that contain EPGs.

Filters

The filter element starts with a `<vzFilter>` tag and contains elements that are indicated with a `<vzEntry>` tag.

The following example defines "Http" and "Https" filters. The first attribute of the filter is its name and the value of the name attribute is a string that is unique to the tenant. These names can be reused in different tenants. These filters are used in the subject elements within contracts later on in the example.

```
<vzFilter name="Http">
  <vzEntry name="e1" etherT="ipv4" prot="tcp" dFromPort="80" dToPort="80"/>
</vzFilter>

<vzFilter name="Https">
  <vzEntry name="e1" etherT="ipv4" prot="tcp" dFromPort="443" dToPort="443"/>
</vzFilter>
```

This example defines these two filters: Http and Https. The first attribute of the filter is its name and the value of the name attribute is a string that is unique to the tenant, i.e. these names can be reused in different tenants. These filters will be used in the subject elements within contracts later on in the example.

Each filter can have one or more entries where each entry describes a set of Layer 4 TCP or UDP port numbers. Some of the possible attributes of the `<vzEntry>` element are as follows:

- name
- prot
- dFromPort
- dToPort
- sFromPort
- sToPort
- etherT
- ipFlags
- arpOpc

- tcpRules

In this example, each entry's name attribute is specified. The name is an ASCII string that must be unique within the filter but can be reused in other filters. Because this example does not refer to a specific entry later on, it is given a simple name of "e1".

The EtherType attribute, `etherT`, is next. It is assigned the value of `ipv4` to specify that the filter is for IPv4 packets. There are many other possible values for this attribute. Common ones include `ARP`, `RARP`, and `IPv6`. The default is `unspecified` so it is important to assign it a value.

Following the EtherType attribute is the `prot` attribute that is set to `tcp` to indicate that this filter is for TCP traffic. Alternate protocol attributes include `udp`, `icmp`, and `unspecified` (default).

After the protocol, the destination TCP port number is assigned to be in the range from 80 to 80 (exactly TCP port 80) with the `dFromPort` and `dToPort` attributes. If the from and to are different, they specify a range of port numbers.

In this example, these destination port numbers are specified with the attributes `dFromPort` and `dToPort`. However, when they are used in the contract, they should be used for the destination port from the TCP client to the server and as the source port for the return traffic. See the attribute `revFltPorts` later in this example for more information.

The second filter does essentially the same thing, but for port 443 instead.

Filters are referred to by subjects within contracts by their target distinguished name, `tDn`. The `tDn` name is constructed as follows:

```
uni/tn-<tenant name>/flt-<filter name>
```

For example, the `tDn` of the first filter above is `uni/tn-coke/flt-Http`. The second filter has the name `uni/tn-coke/flt-Https`. In both cases, `solar` comes from the tenant name.

Contracts

The contract element is tagged `vzBrCP` and it has a `name` attribute.

```
<vzBrCP name="webCtrect">
  <vzSubj name="http" revFltPorts="true" provmatchT="All">
    <vzRsSubjFiltAtt tnVzFilterName="Http"/>
    <vzRsSubjGraphAtt graphName="G1" termNodeName="TProv"/>
    <vzProvSubjLbl name="openProv"/>
    <vzConsSubjLbl name="openCons"/>
  </vzSubj>
  <vzSubj name="https" revFltPorts="true" provmatchT="All">
    <vzProvSubjLbl name="secureProv"/>
    <vzConsSubjLbl name="secureCons"/>
    <vzRsFiltAtt tnVzFilterName="Https "/>
    <vzRsOutTermGraphAtt graphName="G2" termNodeName="TProv"/>
  </vzSubj>
</vzBrCP>
```

Contracts are the policy elements between EPGs. They contain all of the filters that are applied between EPGs that produce and consume the contract. The contract element is tagged `vzBrCP` and it has a `name` attribute. Refer to the object model reference documentation for other attributes that can be used in the contract element. This example has one contract named `webCtrect`.

The contract contains multiple subject elements where each subject contains a set of filters. In this example, the two subjects are `http` and `https`.

The contract is later referenced by EPGs that either provide or consume it. They reference it by its name in the following manner:

```
uni/tn-[tenant-name]/brc-[contract-name]
```

`tenant-name` is the name of the tenant, “solar” in this example, and the `contract-name` is the name of the contract. For this example, the `tDn` name of the contract is `uni/tn-solar/brc-webCtrct`.

Subjects

The subject element starts with the tag `vzSubj` and has three attributes: `name`, `revFltPorts`, and `matchT`. The `name` is simply the ASCII name of the subject.

`revFltPorts` is a flag that indicates that the Layer 4 source and destination ports in the filters of this subject should be used as specified in the filter description in the forward direction (that is, in the direction of from consumer to producer EPG), and should be used in the opposite manner for the reverse direction. In this example, the “http” subject contains the “Http” filter that defined TCP destination port 80 and did not specify the source port. Because the `revFltPorts` flag is set to true, the policy will be TCP **destination port 80** and any source port for traffic from the consumer to the producer, and it will be TCP destination port any and **source port 80** for traffic from the producer to the consumer. The assumption is that the consumer initiates the TCP connection to the producer (the consumer is the client and the producer is the server).

The default value for the `revFltPrts` attribute is `false` if it is not specified.

Labels

The match type attribute, `provmatchT` (for provider matching) and `consmatchT` (for consumer matching) determines how the subject labels are compared to determine if the subject applies for a given pair of consumers and producers. The following match type values are available:

- All
- AtLeastOne (default)
- None
- ExactlyOne

When deciding whether a subject applies to the traffic between a producer and consumer EPG, the `match` attribute determines how the subject labels that are defined (or not) in those EPGs should be compared to the labels in the subject. If the `match` attribute value is set to `All`, it only applies to the providers whose provider subject labels, `vzProvSubjLbl`, match all of the `vzProvSubjLbl` labels that are defined in the subject. If two labels are defined, both must also be in the provider. If a provider EPG has 10 labels, as long as all of the provider labels in the subject are present, a match is confirmed. A similar criteria is used for the consumers that use the `vzConsSubjLbl`. If the `matchT` attribute value is `AtLeastOne`, only one of the labels must match. If the `matchT` attribute is `None`, the match only occurs if none of the provider labels in the subject match the provider labels of the provider EPGs and similarly for the consumer.

If the producer or consumer EPGs do not have any subject labels and the subject does not have any labels, a match occurs for `All`, `AtLeastOne`, and `None` (if you do not use labels, the subject is used and the `matchT` attribute does not matter).

An optional attribute of the subject not shown in the example is `prio` where the priority of the traffic that matches the filter is specified. Possible values are `gold`, `silver`, `bronze`, or `unspecified` (default).

In the example, the subject element contains references to filter elements, subject label elements, and graph elements. `<vzRsSubjFiltAtt tDn="uni/tn-coke/flt-Http"/>` is a reference to a previously defined filter. This element is identified by the `vzRsSubjFiltAtt` tag.

`<vzRsSubjGraphAtt graphName="G1" termNodeName="TProv"/>` defines a terminal connection.

`<vzProvSubjLbl name="openProv"/>` defines a provider label named "openProv". The label is used to qualify or filter which subjects get applied to which EPGs. This particular one is a provider label and the corresponding consumer label is identified by the tag `vzConsSubjLbl`. These labels are matched with the corresponding label of the provider or consumer EPG that is associated with the current contract. If a match occurs according to the `matchT` criteria described above, a particular subject applies to the EPG. If no match occurs, the subject is ignored.

Multiple provider and consumer subject labels can be added to a subject to allow more complicated matching criteria. In this example, there is just one label of each type on each subject. However, the labels on the first subject are different from the labels on the second subject, which allows these two subjects to be handled differently depending on the labels of the corresponding EPGs. The order of the elements within the subject element does not matter.

VRF

The Virtual Routing and Forwarding (VRF) (also known as a context or private network) is identified by the `fvCtx` tag and contains a name attribute.

A tenant can contain multiple VRFs. For this example, the tenant uses one VRF named "solartx1". The name must be unique within the tenant.

The VRF defines a Layer 3 address domain. All of the endpoints within the Layer 3 domain must have unique IPv4 or IPv6 addresses, because it is possible to directly forward packets between these devices if the policy allows it.

Although a VRF defines a unique IP address space, the corresponding subnets are defined within bridge domains. Each bridge domain is then associated with the VRF.

Bridge Domains

The bridge domain element is identified with the `fvBD` tag and has a name attribute.

```
<fvBD name="solarBD1">
  <fvRsCtx tnFvCtxName="solarctx1" />
  <fvSubnet ip="11.22.22.20/24">
    <fvRsBDSubnetToProfile
      tnL3extOutName="rout1"
      tnRtctrlProfileName="profExport" />
  </fvSubnet>
  <fvSubnet ip="11.22.23.211/24">
    <fvRsBDSubnetToProfile
      tnL3extOutName="rout1"
      tnRtctrlProfileName="profExport"/>
  </fvSubnet>
</fvBD>
```

Within the bridge domain element, subnets are defined and a reference is made to the corresponding Virtual Routing and Forwarding (VRF) instance (also known as a context or private network). Each bridge domain must be linked to a VRF and have at least one subnet.

This example uses one bridge domain named “solarBD1”. In this example, the “solarctx1” VRF is referenced by using the element tagged `fvRsCtx` and the `tnFvCtxName` attribute is given the value “solarctx1”. This name comes from the VRF defined above.

The subnets are contained within the bridge domain and a bridge domain can contain multiple subnets. This example defines two subnets. All of the addresses used within a bridge domain must fall into one of the address ranges that is defined by the subnets. However, the subnet can also be a supernet which is a very large subnet that includes many addresses that might never be used. Specifying one giant subnet that covers all current future addresses can simplify the bridge domain specification. However, different subnets must not overlap within a bridge domain or with subnets defined in other bridge domains that are associated with the same VRF. Subnets can overlap with other subnets that are associated with other VRFs.

The subnets described above are 11.22.22.xx/24 and 11.22.23.xx/24. However, the full 32 bits of the address is given even though the mask says that only 24 are used, because this IP attribute also identifies the full IP address for the router in that subnet. In the first case, the router IP address (default gateway) is 11.22.22.20 and for the second subnet, it is 11.22.23.211.

The entry 11.22.22.20/24 is equivalent to the following, but in compact form:

- Subnet: 11.22.22.00
- Subnet Mask: 255.255.255.0
- Default gateway: 11.22.22.20

Application Profiles

The start of the application profile is indicated by the `fvAp` tag and has a name attribute.

```
<fvAp name="sap">
```

This example has one application network profile and it is named “sap.”

The application profile is a container that holds the EPGs. EPGs can communicate with other EPGs in the same application profile and with EPGs in other application profiles. The application profile is simply a convenient container that is used to hold multiple EPGs that are logically related to one another. They can be organized by the application they provide such as “sap,” by the function they provide such as “infrastructure,” by where they are in the structure of the data center such as “DMZ,” or whatever organizing principle the administrator chooses to use.

The primary object that the application profile contains is an endpoint group (EPG). In this example, the “sap” application profile contains 4 EPGs: web1, web2, app, and db.

Endpoints and Endpoint Groups (EPGs)

EPGs begin with the tag `fvAEPg` and have a name attribute.

```
<fvAEPg name="web1">
  <fvRsBd tnFvBDName="solarBD1" />
  <fvRsDomAtt tDn="uni/vmmp-VMware/dom-mininet" />
  <fvRsProv tnVzBrCPName="webCtrct" matchT="All">
    <vzProvSubjLbl name="openProv"/>
    <vzProvSubjLbl name="secureProv"/>
    <vzProvLbl name="green"/>
  </fvRsProv>
</fvAEPg>
```

The EPG is the most important fundamental object in the policy model. It represents a collection of endpoints that are treated in the same fashion from a policy perspective. Rather than configure and manage those endpoints individually, they are placed within an EPG and are managed as a collection or group.

The EPG object is where labels are defined that govern what policies are applied and which other EPGs can communicate with this EPG. It also contains a reference to the bridge domain that the endpoints within the EPG are associated with as well as which virtual machine manager (VMM) domain they are associated with. VMM allows virtual machine mobility between two VM servers instantaneously with no application downtime.

The first EPG in the example is named “web1.” The `fvRsBd` element within the EPG defines which bridge domain that it is associated with. The bridge domain is identified by the value of the `tnFvBDName` attribute. This EPG is associated with the “solarBD1” bridge domain named in the “Bridge Domain” section above. The binding to the bridge domain is used by the system to understand what the default gateway address should be for the endpoints in this EPG. It does not imply that the endpoints are all in the same subnet or that they can only communicate through bridging. Whether an endpoint’s packets are bridged or routed is determined by whether the source endpoint sends the packet to its default gateway or the final destination desired. If it sends the packet to the default gateway, the packet is routed.

The VMM domain used by this EPG is identified by the `fvRsDomAtt` tag. This element references the VMM domain object defined elsewhere. The VMM domain object is identified by its `tDn` name attribute. This example shows only one VMM domain called “uni/vmmp-VMware/dom-mininet.”

The next element in the “web1” EPG defines which contract this EPG provides and is identified by the `fvRsProv` tag. If “web1” were to provide multiple contracts, there would be multiple `fvRsProv` elements. Similarly, if it were to consume one or more contracts, there would be `fvRsCons` elements as well.

The `fvRsProv` element has a required attribute that is the name of the contract that is being provided. “web1” is providing the contract “webCtrct” that was defined earlier that was called `tDn=“uni/tn-coke/brc-webCtrct”`.

The next attribute is the `matchT` attribute, which has the same semantics for matching provider or consumer labels as it did in the contract for subject labels (it can take on the values of `All`, `AtLeastOne`, or `None`). This criteria applies to the provider labels as they are compared to the corresponding consumer labels. A match of the labels implies that the consumer and provider can communicate if the contract between them allows it. In other words, the contract has to allow communication and the consumer and provider labels have to match using the match criteria specified at the provider.

The consumer has no corresponding match criteria. The match type used is always determined by the provider.

Inside the provider element, `fvRsProv`, an administrator needs to specify the labels that are to be used. There are two kinds of labels, provider labels and provider subject labels. The provider labels, `vzProvLbl`, are used to match consumer labels in other EPGs that use the `matchT` criteria described earlier. The provider subject labels, `vzProvSubjLbl`, are used to match the subject labels that are specified in the contract. The only attribute of the label is its name attribute.

In the “web1” EPG, two provider subject labels, `openProv` and `secureProv`, are specified to match with the “http” and “https” subjects of the “webCtrct” contract. One provider label, “green” is specified with a match criteria of `All` that will match with the same label in the “App” EPG.

The next EPG in the example, “web2,” is very similar to “web1” except that there is only one `vzProvSubjLbl` and the labels themselves are different.

The third EPG is one called “app” and it is defined as follows:

```
<fvAEPg name="app">
  <fvRsBd tnFvBDName="solarBD1" />
  <fvRsDomAtt tDn="uni/vmmp-VMware/dom-mininet" />
  <fvRsCons tnVzBrCPName="webCtrct">
    <vzConsSubjLbl name="openCons"/>
  </fvRsCons>
</fvAEPg>
```



```

        <vzConsSubjLbl name="secureCons"/>
        <vzConsLbl name="green"/>
    </fvRsCons>
</fvABPp>

```

The first part is nearly the same as the “web1” EPG. The major difference is that this EPG is a consumer of the “webCtrct” and has the corresponding consumer labels and consumer subject labels. The syntax is nearly the same except that “Prov” is replaced by “Cons” in the tags. There is no match attribute in the `FvRsCons` element because the match type for matching the provider with consumer labels is specified in the provider.

In the last EPG, “db” is very similar to the “app” EPG in that it is purely a consumer.

While in this example, the EPGs were either consumers or producers of a single contract, it is typical for an EPG to be at once a producer of multiple contracts and a consumer of multiple contracts.

Closing

```

    </fvAp>
</fvTenant>
</polUni>

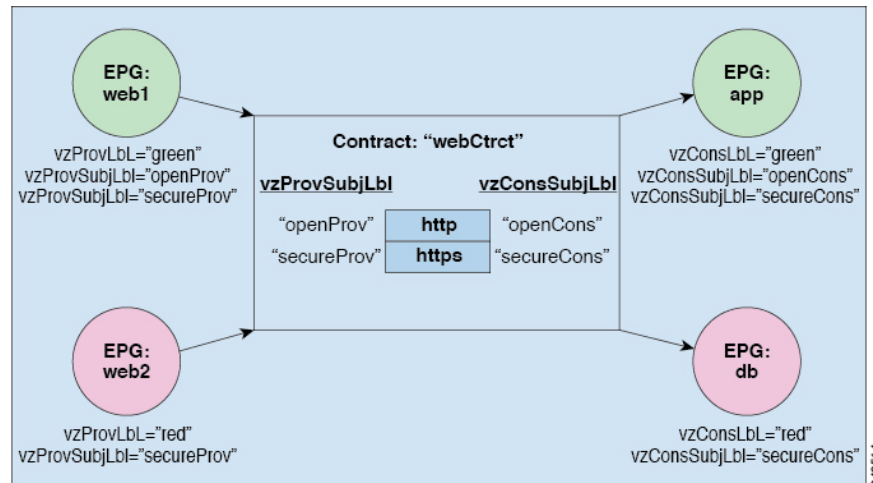
```

The final few lines complete the policy.

What the Example Tenant Policy Does

The following figure shows how contracts govern endpoint group (EPG) communications.

Figure 9: Labels and Contract Determine EPG to EPG Communications



The four EPGs are named EPG:web1, EPG:web2, EPG:app, and EPG:db. EPG:web1 and EPG:web2 provide a contract called webCtrct. EPG:app and EPG:db consume that same contract.

EPG:web1 can only communicate with EPG:app and EPG:web2 can only communicate with EPG:db. This interaction is controlled through the provider and consumer labels “green” and “red”.

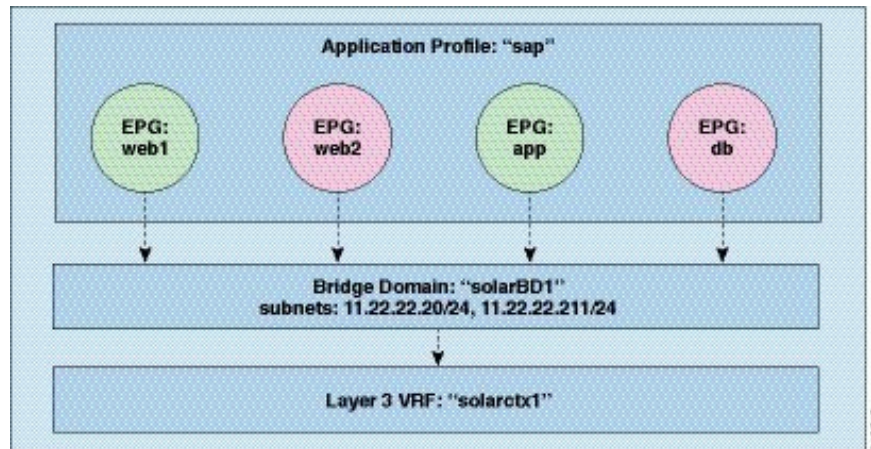
When EPG:web1 communicates with EPG:app, they use the webCtrct contract. EPG:app can initiate connections to EPG:web1 because it consumes the contract that EPG:web1 provides.

The subjects that EPG:web1 and EPG:app can use to communicate are both http and https because EPG:web1 has the provider subject label “openProv” and the http subject also has it. EPG:web1 has the provider subject

label “secureProv” as does the subject https. In a similar fashion, EPG:app has subject labels “openCons” and “secureCons” that subjects http and https have.

When EPG:web2 communicates with EPG:db, they can only use the https subject because only the https subject carries the provider and consumer subject labels. EPG:db can initiate the TCP connection to EPG:web2 because EPG:db consumes the contract provided by EPG:web2.

Figure 10: Bridge Domain, Subnets, and Layer 3 VRF



The example policy specifies the relationship between EPGs, application profiles, bridge domains, and Layer 3 Virtual Routing and Forwarding (VRF) instances in the following manner: the EPGs EPG:web1, EPG:web2, EPG:app, and EPG:db are all members of the application profile called “sap.”

These EPGs are also linked to the bridge domain “solarBD1.” solarBD1 has two subnets, 11.22.22.XX/24 and 11.22.23.XX/24. The endpoints in the four EPGs must be within these two subnet ranges. The IP address of the default gateway in those two subnets will be 11.22.22.20 and 11.22.23.211. The solarBD1 bridge domain is linked to the “solarctx1” Layer 3 VRF.

All these policy details are contained within a tenant called “solar.”

EPGs

Deploying an Application EPG through an AEP or Interface Policy Group to Multiple Ports

Through the APIC Advanced GUI and REST API, you can associate attached entity profiles directly with application EPGs. By doing so, you deploy the associated application EPGs to all those ports associated with the attached entity profile in a single configuration.

Through the APIC REST API or the NX-OS style CLI, you can deploy an application EPG to multiple ports through an Interface Policy Group.

Deploying an EPG on a Specific Port with APIC Using the REST API

Before you begin

The tenant where you deploy the EPG is created.

Deploy an EPG on a specific port.

Example:

```
<fvTenant name="<tenant_name>" dn="uni/tn-test1" >
  <fvCtx name="<network_name>" pcEnfPref="enforced" knwMcastAct="permit"/>
  <fvBD name="<bridge_domain_name>" unkMcastAct="flood" >
    <fvRsCtx tnFvCtxName="<network_name>"/>
  </fvBD>
  <fvAp name="<application_profile>" >
    <fvAEPg name="<epg_name>" >
      <fvRsPathAtt tDn="topology/pod-1/paths-1017/pathep-[eth1/13]" mode="regular"
instrImedcy="immediate" encap="vlan-20"/>
    </fvAEPg>
  </fvAp>
</fvTenant>
```

Deploying an EPG through an AEP to Multiple Interfaces Using the REST API

The interface selectors in the AEP enable you to configure multiple paths for an AEPg. The following can be selected:

1. A node or a group of nodes
2. An interface or a group of interfaces
The interfaces consume an interface policy group (and so an `infra:AttEntityP`).
3. The `infra:AttEntityP` is associated to the AEPg, thus specifying the VLANs to use.
An `infra:AttEntityP` can be associated with multiple AEPgs with different VLANs.

When you associate the `infra:AttEntityP` with the AEPg, as in 3, this deploys the AEPg on the nodes selected in 1, on the interfaces in 2, with the VLAN provided by 3.

In this example, the AEPg `uni/tn-Coke/ap-AP/epg-EPG1` is deployed on interfaces 1/10, 1/11, and 1/12 of nodes 101 and 102, with `vlan-102`.

Before you begin

- Create the target application EPG (AEPg).
- Create the VLAN pool containing the range of VLANs you wish to use for EPG deployment with the Attached Entity Profile (AEP).
- Create the physical domain and link it to the VLAN pool and AEP.

To deploy an AEPg on selected nodes and interfaces, send a post with XML such as the following:

Example:

```
<infraInfra dn="uni/infra">
  <infraNodeP name="NodeProfile">
    <infraLeafS name="NodeSelector" type="range">
      <infraNodeBlk name="NodeBlk" from="101" to="102"/>
      <infraRsAccPortP tDn="uni/infra/accportprof-InterfaceProfile"/>
    </infraLeafS>
  </infraNodeP>

  <infraAccPortP name="InterfaceProfile">
    <infraHPortS name="InterfaceSelector" type="range">
      <infraPortBlk name="InterfaceBlock" fromCard="1" toCard="1" fromPort="10" toPort="12"/>
      <infraRsAccBaseGrp tDn="uni/infra/funcprof/accportgrp-PortGrp" />
    </infraHPortS>
  </infraAccPortP>

  <infraFuncP>
    <infraAccPortGrp name="PortGrp">
      <infraRsAttEntP tDn="uni/infra/attentp-AttEntityProfile"/>
    </infraAccPortGrp>
  </infraFuncP>

  <infraAttEntityP name="AttEntityProfile" >
    <infraGeneric name="default" >
      <infraRsFuncToEpg tDn="uni/tn-Coke/ap-AP/epg-EPG1" encap="vlan-102"/>
    </infraGeneric>
  </infraAttEntityP>
</infraInfra>
```

Creating AEP, Domains, and VLANs to Deploy an EPG on a Specific Port Using the REST API

Before you begin

- The tenant where you deploy the EPG is already created.
- An EPG is statically deployed on a specific port.

Step 1 Create the interface profile, switch profile and the Attach Entity Profile (AEP).

Example:

```
<infraInfra>

  <infraNodeP name="<switch_profile_name>" dn="uni/infra/nprof-<switch_profile_name>" >
    <infraLeafS name="SwitchSeletor" descr="" type="range">
      <infraNodeBlk name="nodeBlk1" descr="" to_"1019" from_"1019"/>
    </infraLeafS>
    <infraRsAccPortP tDn="uni/infra/accportprof-<interface_profile_name>"/>
  </infraNodeP>

  <infraAccPortP name="<interface_profile_name>" dn="uni/infra/accportprof-<interface_profile_name>"
```

```

>
  <infraHPortS name="portSelector" type="range">
    <infraRsAccBaseGrp tDn="uni/infra/funcprof/accportgrp-<port_group_name>" fexId="101"/>
    <infraPortBlk name="block2" toPort="13" toCard="1" fromPort="11" fromCard="1"/>
  </infraHPortS>
</infraAccPortP>

<infraAccPortGrp name="<port_group_name>" dn="uni/infra/funcprof/accportgrp-<port_group_name>"
>
  <infraRsAttEntP tDn="uni/infra/attentp-<attach_entity_profile_name>"/>
  <infraRsHifPol tnFabricHifPolName="1GHifPol"/>
</infraAccPortGrp>

  <infraAttEntityP name="<attach_entity_profile_name>"
dn="uni/infra/attentp-<attach_entity_profile_name>" >
    <infraRsDomP tDn="uni/phys-<physical_domain_name>"/>
  </infraAttEntityP>

</infraInfra>

```

Step 2 Create a domain.**Example:**

```

<physDomP name="<physical_domain_name>" dn="uni/phys-<physical_domain_name>">
  <infraRsVlanNs tDn="uni/infra/vlanns-[<vlan_pool_name>]-static"/>
</physDomP>

```

Step 3 Create a VLAN range.**Example:**

```

<fvnsVlanInstP name="<vlan_pool_name>" dn="uni/infra/vlanns-[<vlan_pool_name>]-static"
allocMode="static">
  <fvnsEncapBlk name="" descr="" to="vlan-25" from="vlan-10"/>
</fvnsVlanInstP>

```

Step 4 Associate the EPG with the domain.**Example:**

```

<fvTenant name="<tenant_name>" dn="uni/tn-" >
  <fvAEPg prio="unspecified" name="<epg_name>" matchT="AtleastOne"
dn="uni/tn-test1/ap-AP1/epg-<epg_name>" descr="">
    <fvRsDomAtt tDn="uni/phys-<physical_domain_name>" instrImedcy="immediate"
resImedcy="immediate"/>
  </fvAEPg>
</fvTenant>

```

Intra-EPG Isolation

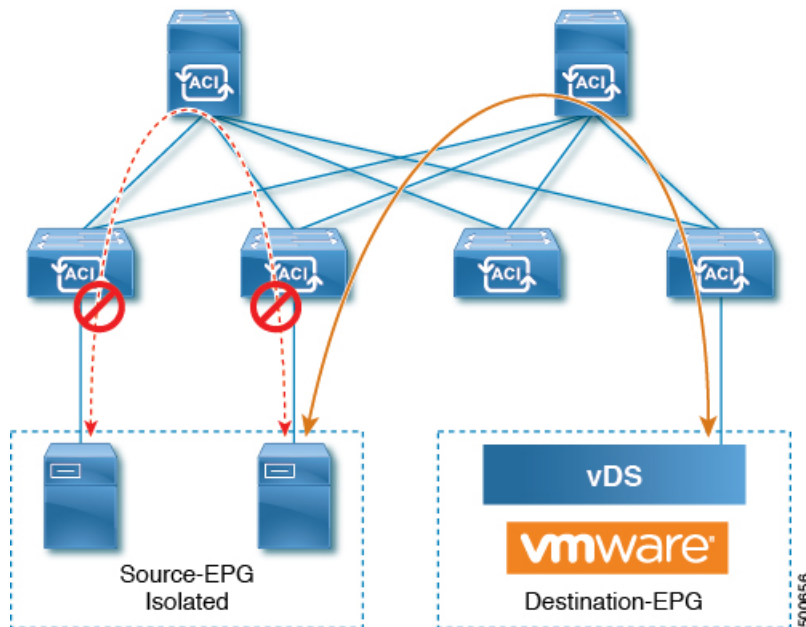
Intra-EPG Isolation for Bare Metal Servers

Intra-EPG endpoint isolation policies can be applied to directly connected endpoints such as bare metal servers.

Examples use cases include the following:

- Backup clients have the same communication requirements for accessing the backup service, but they don't need to communicate with each other.
- Servers behind a load balancer have the same communication requirements, but isolating them from each other protects against a server that is compromised or infected.

Figure 11: Intra-EPG Isolation for Bare Metal Servers



Bare metal EPG isolation is enforced at the leaf switch. Bare metal servers use VLAN encapsulation. All unicast, multicast and broadcast traffic is dropped (denied) within isolation enforced EPGs. ACI bridge-domains can have a mix of isolated and regular EPGs. Each Isolated EPG can have multiple VLANs where intra-vlan traffic is denied.

Configuring Intra-EPG Isolation for Bare Metal Servers Using the REST API

Before you begin

The port the EPG uses must be associated with a bare metal server interface in the physical domain.

SUMMARY STEPS

1. Send this HTTP POST message to deploy the application using the XML API.
2. Include this XML structure in the body of the POST message.

DETAILED STEPS

Step 1 Send this HTTP POST message to deploy the application using the XML API.

Example:

```
POST https://apic-ip-address/api/mo/uni/tn-ExampleCorp.xml
```

Step 2 Include this XML structure in the body of the POST message.

Example:

```
<fvTenant name="Tenant_BareMetal" >
  <fvAp name="Web">
    <fvAEPg name="IntraEPGDeny" pcEnfPref="enforced">
      <!-- pcEnfPref="enforced" ENABLES ISOLATION-->
      <fvRsBd tnFvBDName="bd" />
      <fvRsDomAtt tDn="uni/phys-Dom1" />
      <!-- PATH ASSOCIATION -->
      <fvRsPathAtt tDn="topology/pod-1/paths-1017/pathep-[eth1/2]" encap="vlan-51"
primaryEncap="vlan-100" instrImedcy='immediate' />
    </fvAEPg>
  </fvAp>
</fvTenant>
```

Intra-EPG Isolation for VMware VDS or Microsoft Hyper-V Virtual Switch

Intra-EPG Isolation is an option to prevent physical or virtual endpoint devices that are in the same base EPG or uSeg EPG from communicating with each other. By default, endpoint devices included in the same EPG are allowed to communicate with one another. However, conditions exist in which total isolation of the endpoint devices from one another within an EPG is desirable. For example, you may want to enforce intra-EPG isolation if the endpoint VMs in the same EPG belong to multiple tenants, or to prevent the possible spread of a virus.

A Cisco ACI virtual machine manager (VMM) domain creates an isolated PVLAN port group at the VMware VDS or Microsoft Hyper-V Virtual Switch for each EPG that has intra-EPG isolation enabled. A fabric administrator specifies primary encapsulation or the fabric dynamically specifies primary encapsulation at the time of EPG-to-VMM domain association. When the fabric administrator selects the VLAN-pri and VLAN-sec values statically, the VMM domain validates that the VLAN-pri and VLAN-sec are part of a static block in the domain pool.

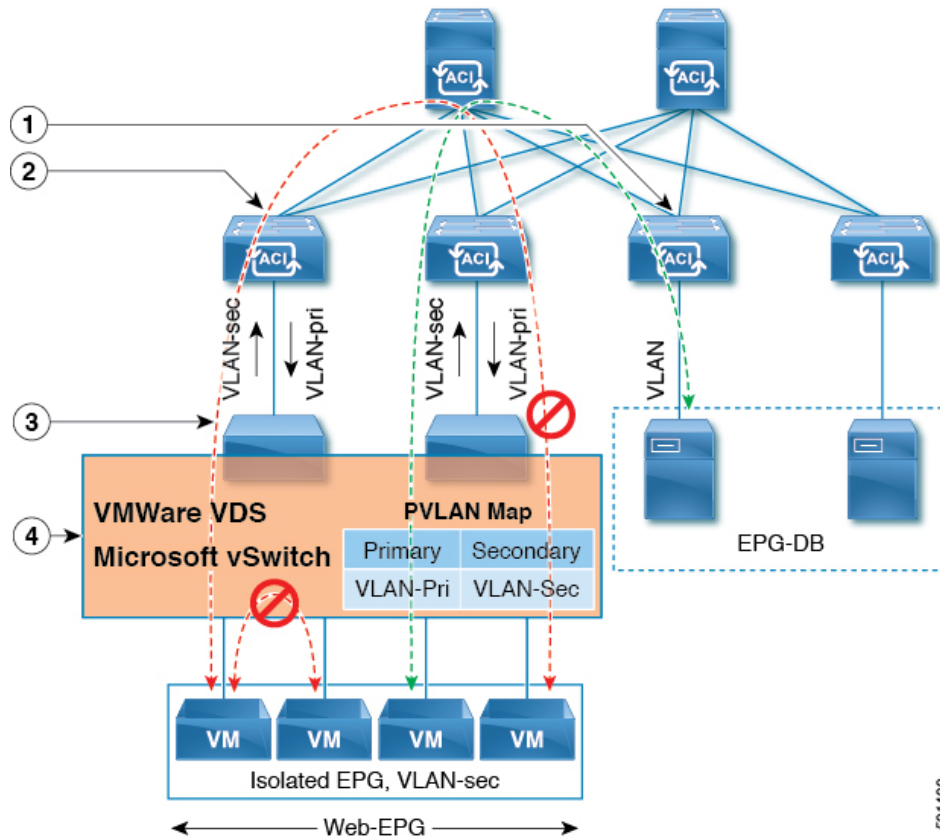


Note When intra-EPG isolation is not enforced, the VLAN-pri value is ignored even if it is specified in the configuration.

VLAN-pri/VLAN-sec pairs for the VMware VDS or Microsoft Hyper-V Virtual Switch are selected per VMM domain during the EPG-to-domain association. The port group created for the intra-EPG isolation EPGs uses the VLAN-sec tagged with type set to PVLAN. The VMware VDS or the Microsoft Hyper-V Virtual Switch and fabric swap the VLAN-pri/VLAN-sec encapsulation:

- Communication from the Cisco ACI fabric to the VMware VDS or Microsoft Hyper-V Virtual Switch uses VLAN-pri.
- Communication from the VMware VDS or Microsoft Hyper-V Virtual Switch to the Cisco ACI fabric uses VLAN-sec.

Figure 12: Intra-EPG Isolation for VMware VDS or Microsoft Hyper-V Virtual Switch



Note these details regarding this illustration:

1. EPG-DB sends VLAN traffic to the Cisco ACI leaf switch. The Cisco ACI egress leaf switch encapsulates traffic with a primary VLAN (PVLAN) tag and forwards it to the Web-EPG endpoint.
2. The VMware VDS or Microsoft Hyper-V Virtual Switch sends traffic to the Cisco ACI leaf switch using VLAN-sec. The Cisco ACI leaf switch drops all intra-EPG traffic because isolation is enforced for all intra VLAN-sec traffic within the Web-EPG.
3. The VMware VDS or Microsoft Hyper-V Virtual Switch VLAN-sec uplink to the Cisco ACI Leaf is in isolated trunk mode. The Cisco ACI leaf switch uses VLAN-pri for downlink traffic to the VMware VDS or Microsoft Hyper-V Virtual Switch.
4. The PVLAN map is configured in the VMware VDS or Microsoft Hyper-V Virtual Switch and Cisco ACI leaf switches. VM traffic from WEB-EPG is encapsulated in VLAN-sec. The VMware VDS or Microsoft Hyper-V Virtual Switch denies local intra-WEB EPG VM traffic according to the PVLAN tag. All intra-ESXi host or Microsoft Hyper-V host VM traffic is sent to the Cisco ACI leaf using VLAN-Sec.

Related Topics

For information on configuring intra-EPG isolation in a Cisco AVS environment, see [Intra-EPG Isolation Enforcement for Cisco AVS, on page 176](#).

Configuring Intra-EPG Isolation for VMware VDS or Microsoft Hyper-V Virtual Switch using the REST API

SUMMARY STEPS

1. Send this HTTP POST message to deploy the application using the XML API.
2. For a VMware VDS or Microsoft Hyper-V Virtual Switch deployment, include one of the following XML structures in the body of the POST message.

DETAILED STEPS

Step 1 Send this HTTP POST message to deploy the application using the XML API.

Example:

```
POST https://apic-ip-address/api/mo/uni/tn-ExampleCorp.xml
```

Step 2 For a VMware VDS or Microsoft Hyper-V Virtual Switch deployment, include one of the following XML structures in the body of the POST message.

Example:

The following example is for VMware VDS:

```
<fvTenant name="Tenant_VMM" >
  <fvAp name="Web">
    <fvAEPg name="IntraEPGDeny" pcEnfPref="enforced">
      <!-- pcEnfPref="enforced" ENABLES ISOLATION-->
      <fvRsBd tnFvBDName="bd" />
      <!-- STATIC ENCAP ASSOCIATION TO VMM DOMAIN-->
      <fvRsDomAtt encap="vlan-2001" instrImedcy="lazy" primaryEncap="vlan-2002"
resImedcy="immediate" tDn="uni/vmmp-VMware/dom-DVS1">
    </fvAEPg>
  </fvAp>
</fvTenant>
```

Example:

The following example is for Microsoft Hyper-V Virtual Switch:

```
<fvTenant name="Tenant_VMM" >
  <fvAp name="Web">
    <fvAEPg name="IntraEPGDeny" pcEnfPref="enforced">
      <!-- pcEnfPref="enforced" ENABLES ISOLATION-->
      <fvRsBd tnFvBDName="bd" />
      <!-- STATIC ENCAP ASSOCIATION TO VMM DOMAIN-->
      <fvRsDomAtt tDn="uni/vmmp-Microsoft/dom-domain1">
    <fvRsDomAtt encap="vlan-2004" instrImedcy="lazy" primaryEncap="vlan-2003"
resImedcy="immediate" tDn="uni/vmmp-Microsoft/dom-domain2">
    </fvAEPg>
  </fvAp>
</fvTenant>
```

Intra-EPG Isolation Enforcement for Cisco AVS

By default, endpoints with an EPG can communicate with each other without any contracts in place. However, you can isolate endpoints within an EPG from each other. In some instances, you might want to enforce endpoint isolation within an EPG to prevent a VM with a virus or other problem from affecting other VMs in the EPG.

You can configure isolation on all or none of the endpoints within an application EPG; you cannot configure isolation on some endpoints but not on others.

Isolating endpoints within an EPG does not affect any contracts that enable the endpoints to communicate with endpoints in another EPG.

Isolating endpoints within an EPG will trigger a fault when the EPG is associated with Cisco AVS domains in VLAN mode.



Note Using intra-EPG isolation on a Cisco AVS microsegment (uSeg) EPG is not currently supported. Communication is possible between two endpoints that reside in separate uSeg EPGs if either has intra-EPG isolation enforced, regardless of any contract that exists between the two EPGs.

Configuring Intra-EPG Isolation for Cisco AVS Using the REST API

Before you begin

Make sure that Cisco AVS is in VXLAN mode.

Step 1 Send this HTTP POST message to deploy the application using the XML API.

Example:

```
POST
https://192.0.20.123/api/mo/uni/tn-ExampleCorp.xml
```

Step 2 For a VMM deployment, include the XML structure in the following example in the body of the POST message.

Example:

```
Example:
<fvTenant name="Tenant_VMM" >
  <fvAp name="Web">
    <fvAEPg name="IntraEPGDeny" pcEnfPref="enforced">
      <!-- pcEnfPref="enforced" ENABLES ISOLATION-->
      <fvRsBd tnFvBDName="bd" />
      <fvRsDomAtt encap="vlan-2001" tDn="uni/vmmp-VMware/dom-DVS1">
    </fvAEPg>
  </fvAp>
</fvTenant>
```

What to do next

You can select statistics and view them to help diagnose problems involving the endpoint. See the sections "Choosing Statistics to View for Isolated Endpoints" and "Viewing Statistics for Isolated Endpoints" in the *Cisco AVS Configuration Guide* or the *Cisco APIC Layer 2 Configuration Guide*.

Microsegmentation

Using Microsegmentation with Network-based Attributes on Bare Metal

You can use Cisco APIC to configure Microsegmentation with Cisco ACI to create a new, attribute-based EPG using a network-based attribute, a MAC address or one or more IP addresses. You can configure Microsegmentation with Cisco ACI using network-based attributes to isolate VMs or physical endpoints within a single base EPG or VMs or physical endpoints in different EPGs.

Using an IP-based Attribute

You can use an IP-based filter to isolate a single IP address, a subnet, or multiple of noncontiguous IP addresses in a single microsegment. You might want to isolate physical endpoints based on IP addresses as a quick and simple way to create a security zone, similar to using a firewall.

Using a MAC-based Attribute

You can use a MAC-based filter to isolate a single MAC address or multiple MAC addresses. You might want to do this if you have a server sending bad traffic into the network. By creating a microsegment with a MAC-based filter, you can isolate the server.

Configuring an IP-based Microsegmented EPG as a Shared Resource Using the REST API

You can configure a microsegmented EPG with an IP-Address with 32 bit mask as a shared service, accessible by clients outside of the VRF and the current fabric.

SUMMARY STEPS

1. To configure an IP address-attribute microsegmented EPG `epg3` with a shared subnet, with an IP address and 32-bit mask, send a post with XML such as the following example. In the IP attributes, the attribute `usefvSubnet` is set to "yes."

DETAILED STEPS

To configure an IP address-attribute microsegmented EPG `epg3` with a shared subnet, with an IP address and 32-bit mask, send a post with XML such as the following example. In the IP attributes, the attribute `usefvSubnet` is set to "yes."

Example:

```
<fvAEPg descr="" dn="/uni/tn-t0/ap-a0/epg-epg3" fwdCtrl=""
  isAttrBasedEPg="yes" matchT="AtleastOne" name="epg3" pcEnfPref="unenforced"
  prefGrMemb="exclude"prio="unspecified">
```

```

    <fvRsCons prio="unspecified" tnVzBrCPName="ip-epg"/>
    <fvRsNodeAtt descr="" encap="unknown" instrImedcy="immediate" mode="regular"
tDn="topology/pod-2/node-106"/>
    <fvSubnet ctrl="" descr="" ip="56.4.0.2/32" name="" preferred="no"
      scope="public,shared" virtual="no"/>
    <fvRsDomAtt classPref="encap" delimiter="" encap="unknown" encapMode="auto"
instrImedcy="immediate"
      primaryEncap="unknown" resImedcy="immediate" tDn="uni/phys-vpc"/>
    <fvRsCustQosPol tnQosCustomPolName=""/>
    <fvRsBd tnFvBDName="b2"/>
    <fvCrtrn descr="" match="any" name="default" ownerKey="" ownerTag="" prec="0">
      <fvIpAttr descr="" ip="1.1.1.3" name="ipv4" ownerKey="" ownerTag="" usefvSubnet="yes"/>
    </fvCrtrn>
    <fvRsProv matchT="AtleastOne" prio="unspecified" tnVzBrCPName="ip-epg"/>
    <fvRsProv matchT="AtleastOne" prio="unspecified" tnVzBrCPName="shared-svc"/>
  </fvAEPg>

```

Configuring a Network-Based Microsegmented EPG in a Bare-Metal Environment Using the REST API

This section describes how to configure network attribute microsegmentation with Cisco ACI in a bare-metal environment using the REST API.

SUMMARY STEPS

1. Log in to the Cisco APIC.
2. Post the policy to <https://apic-ip-address/api/node/mo/.xml>.

DETAILED STEPS

- Step 1** Log in to the Cisco APIC.
- Step 2** Post the policy to <https://apic-ip-address/api/node/mo/.xml>.

Example:

A: The following example configures a microsegment named 41-subnet using an IP-based attribute.

```

<polUni>
  <fvTenant dn="uni/tn-User-T1" name="User-T1">
    <fvAp dn="uni/tn-User-T1/ap-Base-EPG" name="Base-EPG">
      <fvAEPg dn="uni/tn-User-T1/ap-Base-EPG/epg-41-subnet" name="41-subnet" pcEnfPref="enforced"
isAttrBasedEPg="yes" >
        <fvRsBd tnFvBDName="BD1" />
        <fvCrtrn name="Security1">
          <fvIpAttr name="41-filter" ip="12.41.0.0/16"/>
        </fvCrtrn>
      </fvAEPg>
    </fvAp>
  </fvTenant>
</polUni>

```

Example:

This example is for base EPG for Example A: .

```

<polUni>
  <fvTenant dn="uni/tn-User-T1" name="User-T1">
    <fvAp dn="uni/tn-User-T1/ap-Base-EPG" name="Base-EPG">
      <fvAEPg dn="uni/tn-User-T1/ap-Base-EPG/baseEPG" name="baseEPG" pcEnfPref="enforced" >
        <fvRsBd tnFvBDName="BD1" />
      </fvAEPg>
    </fvAp>
  </fvTenant>
</polUni>

```

Example:

B: The following example configures a microsegment named useg-epg using a MAC-based attribute.

```

<polUni>
  <fvTenant name="User-T1">
    <fvAp name="customer">
      <fvAEPg name="useg-epg" isAttrBasedEPg="true">
        <fvRsBd tnFvBDName="BD1"/>
        <fvRsDomAtt instrImedcy="immediate" resImedcy="immediate" tDn="uni/phys-phys" />
        <fvRsNodeAtt tDn="topology/pod-1/node-101" instrImedcy="immediate" />
        <fvCrtrn name="default">
          <fvMacAttr name="mac" mac="00:11:22:33:44:55" />
        </fvCrtrn>
      </fvAEPg>
    </fvAp>
  </fvTenant>
</polUni>

```

Configuring Microsegmentation on Virtual Switches

Microsegmentation with the Cisco Application Centric Infrastructure (ACI) provides the ability to automatically assign endpoints to logical security zones called endpoint groups (EPGs) based on various network-based or virtual machine (VM)-based attributes. This section contains instructions for configuring microsegment (uSeg) EPGs on virtual switches.

Microsegmentation with Cisco ACI provides support for virtual endpoints attached to the following:

- VMware vSphere Distributed Switch (VDS)
- Cisco Application Virtual Switch (AVS)
- Microsoft vSwitch

See the [Cisco ACI Virtualization Guide](#) for information about how Microsegmentation with Cisco ACI works, prerequisites, guidelines, and scenarios.

Configuring Microsegmentation with Cisco ACI Using the REST API

This section describes how to configure Microsegmentation with Cisco ACI for Cisco ACI Virtual Edge, Cisco AVS, VMware VDS, or Microsoft Hyper-V Virtual Switch using the REST API.

Step 1 Log in to the Cisco APIC.

Step 2 Post the policy to `https://apic-ip-address/api/node/mo/.xml`.

Example:

This example configures a uSeg EPG with the attributes VM Name containing "vm" and Operating System attributes containing values of "CentOS" and "Linux," with matching for all attributes and with an EPG Match Precedence of 1.

```
<fvAEPg name="Security" isAttrBasedEPg="yes" pcEnfPref="unenforced" status="">
  <fvRsBd tnFvBDName="BD1" />
  <fvRsDomAtt tDn="uni/vmmp-VMware/dom-mininet"/>
  <fvCrtrn name="default" match="all" prec="1">
    <fvVmAttr name="foo" type="vm-name" operator="contains" value="vm"/>
    <fvSCrtrn name="sub-def" match="any">
      <fvVmAttr name="foo1" type="guest-os" operator="contains" value="CentOS"/>
      <fvVmAttr name="foo2" type="guest-os" operator="contains" value="Linux"/>
    </fvSCrtrn>
  </fvCrtrn>
</fvAEPg>
```

Example:

This example is for an application EPG with microsegmentation enabled.

```
<polUni>
  <fvTenant dn="uni/tn-User-T1" name="User-T1">
    <fvAp dn="uni/tn-User-T1/ap-Application-EPG" name="Application-EPG">
      <fvAEPg dn="uni/tn-User-T1/ap-Application-EPG/applicationEPG" name="applicationEPG"
pcEnfPref="enforced" >
        <fvRsBd tnFvBDName="BD1" />
        <fvRsDomAtt tDn="uni/vmmp-VMware/dom-cli-vmm1" classPref="useg"/>
      </fvAEPg>
    </fvAp>
  </fvTenant>
</polUni>
```

In the example above, the string `<fvRsDomAtt tDn="uni/vmmp-VMware/dom-cli-vmm1" classPref="useg"/>` is relevant only for VMware VDS and not for Cisco ACI Virtual Edge, Cisco AVS, or Microsoft Hyper-V Virtual Switch.

Example:

This example attaches a uSeg EPG to a Cisco ACI Virtual Edge VMM domain to add the switching mode.

```
<fvRsDomAtt resImedcyc="immediate" instrImedcyc="immediate" switchingMode="AVE" encapMode="auto"
tDn="uni/vmmp-VMware/dom-AVE-CISCO" primaryEncapInner="" secondaryEncapInner=""/>
```

Application Profiles

Three-Tier Application Deployment

A filter specifies the data protocols to be allowed or denied by a contract that contains the filter. A contract can contain multiple subjects. A subject can be used to realize uni- or bidirectional filters. A unidirectional filter is a filter that is used in one direction, either from consumer-to-provider (IN) or from provider-to-consumer (OUT) filter. A bidirectional filter is the same filter that is used in both directions. It is not reflexive.

Contracts are policies that enable inter-End Point Group (inter-EPG) communication. These policies are the rules that specify communication between application tiers. If no contract is attached to the EPG, inter-EPG

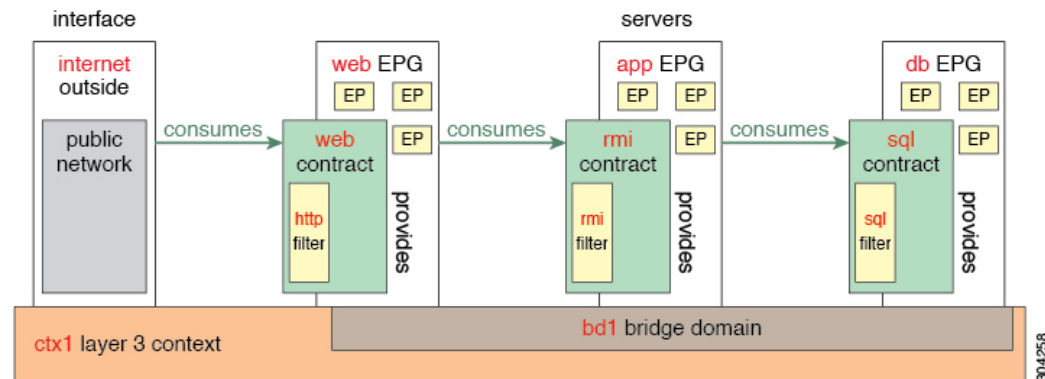
communication is disabled by default. No contract is required for intra-EPG communication because intra-EPG communication is always allowed.

Application profiles enable you to model application requirements that the APIC then automatically renders in the network and data center infrastructure. The application profiles enable administrators to approach the resource pool in terms of applications rather than infrastructure building blocks. The application profile is a container that holds EPGs that are logically related to one another. EPGs can communicate with other EPGs in the same application profile and with EPGs in other application profiles.

To deploy an application policy, you must create the required application profiles, filters, and contracts. Typically, the APIC fabric hosts a three-tier application within a tenant network. In this example, the application is implemented by using three servers (a web server, an application server, and a database server). See the following figure for an example of a three-tier application.

The web server has the HTTP filter, the application server has the Remote Method Invocation (RMI) filter, and the database server has the Structured Query Language (SQL) filter. The application server consumes the SQL contract to communicate with the database server. The web server consumes the RMI contract to communicate with the application server. The traffic enters from the web server and communicates with the application server. The application server then communicates with the database server, and the traffic can also communicate externally.

Figure 13: Three-Tier Application Diagram



Parameters to Create a Filter for http

The parameters to create a filter for http in this example is as follows:

Parameter Name	Filter for http
Name	http
Number of Entries	2
Entry Name	Dport-80 Dport-443
Ethertype	IP
Protocol	tcp tcp

Parameter Name	Filter for http
Destination Port	http https

Parameters to Create Filters for rmi and sql

The parameters to create filters for rmi and sql in this example are as follows:

Parameter Name	Filter for rmi	Filter for sql
Name	rmi	sql
Number of Entries	1	1
Entry Name	Dport-1099	Dport-1521
Ethertype	IP	IP
Protocol	tcp	tcp
Destination Port	1099	1521

Deploying an Application Profile Using the REST API

The port the EPG uses must belong to one of the VM Managers (VMM) or physical domains associated with the EPG.

SUMMARY STEPS

1. Send this HTTP POST message to deploy the application using the XML API.
2. Include this XML structure in the body of the POST message.

DETAILED STEPS

Step 1 Send this HTTP POST message to deploy the application using the XML API.

Example:

```
POST https://apic-ip-address/api/mo/uni/tn-ExampleCorp.xml
```

Step 2 Include this XML structure in the body of the POST message.

Example:

```
<fvTenant name="ExampleCorp">
  <fvAp name="OnlineStore">
    <fvAEPg name="web">
      <fvRsBd tnFvBDName="bd1"/>
      <fvRsCons tnVzBrCPName="rmi"/>
      <fvRsProv tnVzBrCPName="web"/>
      <fvRsDomAtt tDn="uni/vmmp-VMware/dom-datacenter"delimiter=@/>
    </fvAEPg>
  </fvAp>
</fvTenant>
```



```

</fvAEPg>

<fvAEPg name="db">
  <fvRsBd tnFvBDName="bd1"/>
  <fvRsProv tnVzBrCPName="sql"/>
  <fvRsDomAtt tDn="uni/vmmp-VMware/dom-datacenter"/>
</fvAEPg>

<fvAEPg name="app">
  <fvRsBd tnFvBDName="bd1"/>
  <fvRsProv tnVzBrCPName="rmi"/>
  <fvRsCons tnVzBrCPName="sql"/>
  <fvRsDomAtt tDn="uni/vmmp-VMware/dom-datacenter"/>
</fvAEPg>
</fvAp>

<vzFilter name="http" >
<vzEntry dFromPort="80" name="DPort-80" prot="tcp" etherT="ip"/>
<vzEntry dFromPort="443" name="DPort-443" prot="tcp" etherT="ip"/>
</vzFilter>
<vzFilter name="rmi" >
<vzEntry dFromPort="1099" name="DPort-1099" prot="tcp" etherT="ip"/>
</vzFilter>
<vzFilter name="sql">
<vzEntry dFromPort="1521" name="DPort-1521" prot="tcp" etherT="ip"/>
</vzFilter>
  <vzBrCP name="web">
    <vzSubj name="web">
      <vzRsSubjFiltAtt tnVzFilterName="http"/>
    </vzSubj>
  </vzBrCP>

  <vzBrCP name="rmi">
    <vzSubj name="rmi">
      <vzRsSubjFiltAtt tnVzFilterName="rmi"/>
    </vzSubj>
  </vzBrCP>

  <vzBrCP name="sql">
    <vzSubj name="sql">
      <vzRsSubjFiltAtt tnVzFilterName="sql"/>
    </vzSubj>
  </vzBrCP>
</fvTenant>

```

In the string **fvRsDomAtt tDn="uni/vmmp-VMware/dom-datacenter"delimiter=@/, delimiter=@** is optional. If you do not enter a delimiter, the system will use the default | delimiter.

In the XML structure, the first line modifies, or creates if necessary, the tenant named ExampleCorp.

```
<fvTenant name="ExampleCorp">
```

This line creates an application network profile named OnlineStore.

```
<fvAp name="OnlineStore">
```

The elements within the application network profile create three endpoint groups, one for each of the three servers. The following lines create an endpoint group named web and associate it with an existing bridge

domain named bd1. This endpoint group is a consumer, or destination, of the traffic allowed by the binary contract named rmi and is a provider, or source, of the traffic allowed by the binary contract named web. The endpoint group is associated with the VMM domain named datacenter.

```
<fvAEPg name="web">
  <fvRsBd tnFvBDName="bd1"/>
  <fvRsCons tnVzBrCPName="rmi"/>
  <fvRsProv tnVzBrCPName="web"/>
  <fvRsDomAtt tDn="uni/vmmp-VMware/dom-datacenter"/>
</fvAEPg>
```

The remaining two endpoint groups, for the application server and the database server, are created in a similar way.

The following lines define a traffic filter named http that specifies TCP traffic of types HTTP (port 80) and HTTPS (port 443).

```
<vzFilter name="http" >
<vzEntry dFromPort="80" name="DPort-80" prot="tcp" etherT="ip"/>
<vzEntry dFromPort="443" name="DPort-443" prot="tcp" etherT="ip"/>
</vzFilter>
```

The remaining two filters, for application data and database (sql) data, are created in a similar way.

The following lines create a binary contract named web that incorporates the filter named http:

```
<vzBrCP name="web">
  <vzSubj name="web">
    <vzRsSubjFiltAtt tnVzFilterName="http"/>
  </vzSubj>
</vzBrCP>
```

The remaining two contracts, for rmi and sql data protocols, are created in a similar way.

The final line closes the structure:

```
</fvTenant>
```

Contracts, Taboo Contracts, and Preferred Groups

Security Policy Enforcement

As traffic enters the leaf switch from the front panel interfaces, the packets are marked with the EPG of the source EPG. The leaf switch then performs a forwarding lookup on the packet destination IP address within the tenant space. A hit can result in any of the following scenarios:

1. A unicast (/32) hit provides the EPG of the destination endpoint and either the local interface or the remote leaf switch VTEP IP address where the destination endpoint is present.
2. A unicast hit of a subnet prefix (not /32) provides the EPG of the destination subnet prefix and either the local interface or the remote leaf switch VTEP IP address where the destination subnet prefix is present.

- A multicast hit provides the local interfaces of local receivers and the outer destination IP address to use in the VXLAN encapsulation across the fabric and the EPG of the multicast group.



Note Multicast and external router subnets always result in a hit on the ingress leaf switch. Security policy enforcement occurs as soon as the destination EPG is known by the ingress leaf switch.

A miss result in the forwarding table causes the packet to be sent to the forwarding proxy in the spine switch. The forwarding proxy then performs a forwarding table lookup. If it is a miss, the packet is dropped. If it is a hit, the packet is sent to the egress leaf switch that contains the destination endpoint. Because the egress leaf switch knows the EPG of the destination, it performs the security policy enforcement. The egress leaf switch must also know the EPG of the packet source. The fabric header enables this process because it carries the EPG from the ingress leaf switch to the egress leaf switch. The spine switch preserves the original EPG in the packet when it performs the forwarding proxy function.

On the egress leaf switch, the source IP address, source VTEP, and source EPG information are stored in the local forwarding table through learning. Because most flows are bidirectional, a return packet populates the forwarding table on both sides of the flow, which enables the traffic to be ingress filtered in both directions.

Contracts and Taboo Contracts

Contracts Contain Security Policy Specifications

In the ACI security model, contracts contain the policies that govern the communication between EPGs. The contract specifies what can be communicated and the EPGs specify the source and destination of the communications. Contracts link EPGs, as shown below.

EPG 1 ----- CONTRACT ----- EPG 2

Endpoints in EPG 1 can communicate with endpoints in EPG 2 and vice versa if the contract allows it. This policy construct is very flexible. There can be many contracts between EPG 1 and EPG 2, there can be more than two EPGs that use a contract, and contracts can be reused across multiple sets of EPGs, and more.

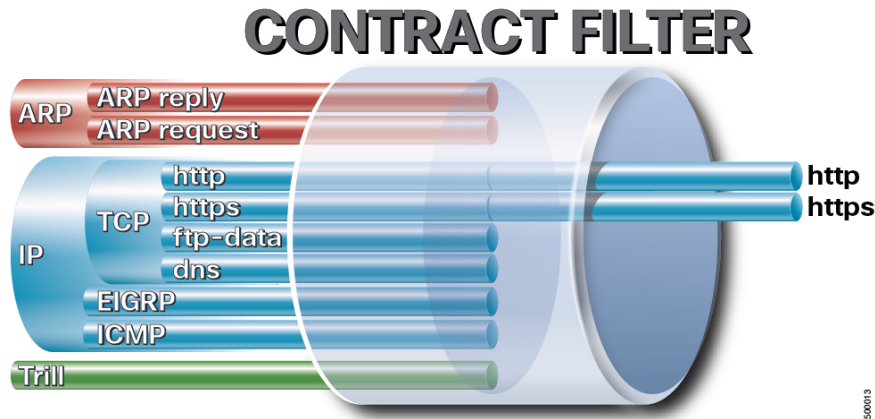
There is also directionality in the relationship between EPGs and contracts. EPGs can either provide or consume a contract. An EPG that provides a contract is typically a set of endpoints that provide a service to a set of client devices. The protocols used by that service are defined in the contract. An EPG that consumes a contract is typically a set of endpoints that are clients of that service. When the client endpoint (consumer) tries to connect to a server endpoint (provider), the contract checks to see if that connection is allowed. Unless otherwise specified, that contract would not allow a server to initiate a connection to a client. However, another contract between the EPGs could easily allow a connection in that direction.

This providing/consuming relationship is typically shown graphically with arrows between the EPGs and the contract. Note the direction of the arrows shown below.

EPG 1 <-----consumes----- CONTRACT <-----provides----- EPG 2

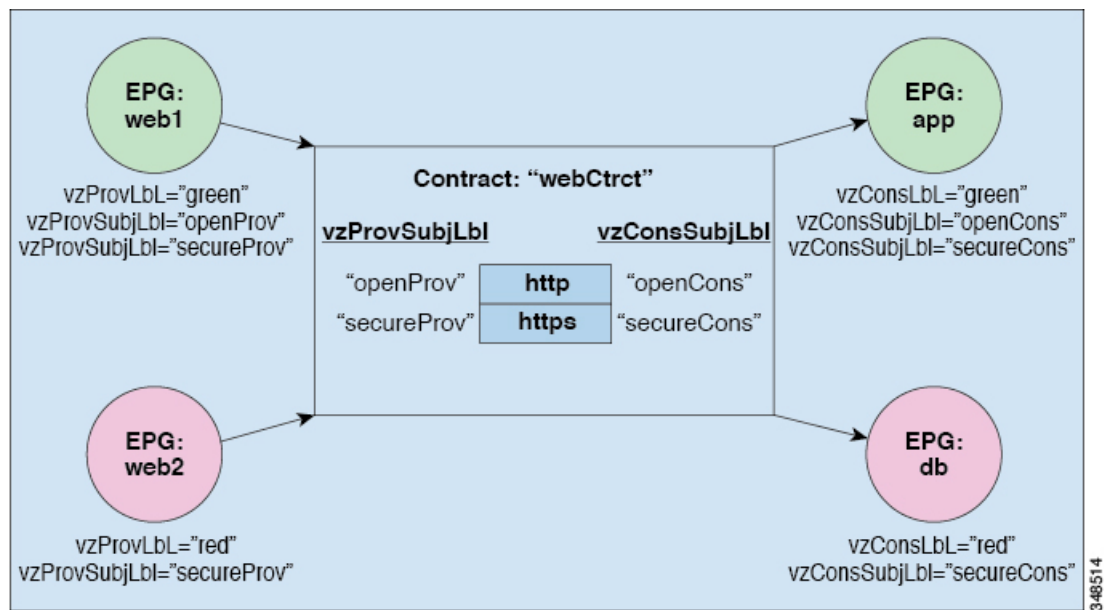
The contract is constructed in a hierarchical manner. It consists of one or more subjects, each subject contains one or more filters, and each filter can define one or more protocols.

Figure 14: Contract Filters



The following figure shows how contracts govern EPG communications.

Figure 15: Contracts Determine EPG to EPG Communications



For example, you may define a filter called HTTP that specifies TCP port 80 and port 8080 and another filter called HTTPS that specifies TCP port 443. You might then create a contract called webCtrct that has two sets of subjects. openProv and openCons are the subjects that contain the HTTP filter. secureProv and secureCons are the subjects that contain the HTTPS filter. This webCtrct contract can be used to allow both secure and non-secure web traffic between EPGs that provide the web service and EPGs that contain endpoints that want to consume that service.

These same constructs also apply for policies that govern virtual machine hypervisors. When an EPG is placed in a virtual machine manager (VMM) domain, the APIC downloads all of the policies that are associated with the EPG to the leaf switches with interfaces connecting to the VMM domain. For a full explanation of VMM domains, see the *Virtual Machine Manager Domains* chapter of *Application Centric Infrastructure Fundamentals*. When this policy is created, the APIC pushes it (pre-populates it) to a VMM domain that specifies which switches allow connectivity for the endpoints in the EPGs. The VMM domain defines the set

of switches and ports that allow endpoints in an EPG to connect to. When an endpoint comes on-line, it is associated with the appropriate EPGs. When it sends a packet, the source EPG and destination EPG are derived from the packet and the policy defined by the corresponding contract is checked to see if the packet is allowed. If yes, the packet is forwarded. If no, the packet is dropped.

Contracts consist of 1 or more subjects. Each subject contains 1 or more filters. Each filter contains 1 or more entries. Each entry is equivalent to a line in an Access Control List (ACL) that is applied on the Leaf switch to which the endpoint within the endpoint group is attached.

In detail, contracts are comprised of the following items:

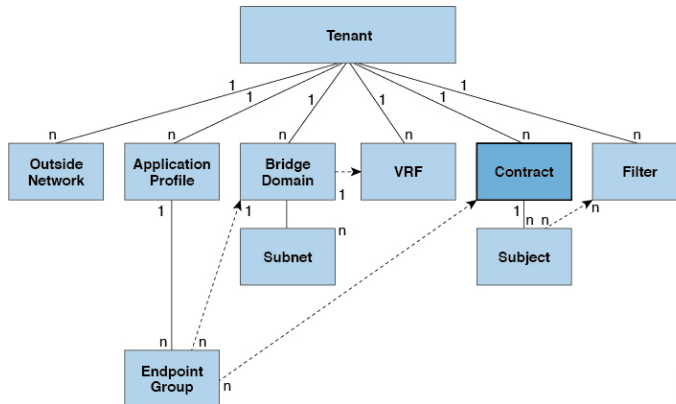
- Name—All contracts that are consumed by a tenant must have different names (including contracts created under the common tenant or the tenant itself).
- Subjects—A group of filters for a specific application or service.
- Filters—Used to classify traffic based upon layer 2 to layer 4 attributes (such as Ethernet type, protocol type, TCP flags and ports).
- Actions—Action to be taken on the filtered traffic. The following actions are supported:
 - Permit the traffic (regular contracts, only)
 - Mark the traffic (DSCP/CoS) (regular contracts, only)
 - Redirect the traffic (regular contracts, only, through a service graph)
 - Copy the traffic (regular contracts, only, through a service graph or SPAN)
 - Block the traffic (taboo contracts)
With Cisco APIC Release 3.2(x) and switches with names that end in EX or FX, you can alternatively use a subject Deny action or Contract or Subject Exception in a standard contract to block traffic with specified patterns.
 - Log the traffic (taboo contracts and regular contracts)
- Aliases—(Optional) A changeable name for an object. Although the name of an object, once created, cannot be changed, the Alias is a property that can be changed.

Thus, the contract allows more complex actions than just allow or deny. The contract can specify that traffic that matches a given subject can be re-directed to a service, can be copied, or can have its QoS level modified. With pre-population of the access policy in the concrete model, endpoints can move, new ones can come on-line, and communication can occur even if the APIC is off-line or otherwise inaccessible. The APIC is removed from being a single point of failure for the network. Upon packet ingress to the ACI fabric, security policies are enforced by the concrete model running in the switch.

Contracts

In addition to EPGs, contracts (`vzBrCP`) are key objects in the policy model. EPGs can only communicate with other EPGs according to contract rules. The following figure shows the location of contracts in the management information tree (MIT) and their relation to other objects in the tenant.

Figure 16: Contracts



An administrator uses a contract to select the type(s) of traffic that can pass between EPGs, including the protocols and ports allowed. If there is no contract, inter-EPG communication is disabled by default. There is no contract required for intra-EPG communication; intra-EPG communication is always implicitly allowed.

You can also configure contract preferred groups that enable greater control of communication between EPGs in a VRF. If most of the EPGs in the VRF should have open communication, but a few should only have limited communication with the other EPGs, you can configure a combination of a contract preferred group and contracts with filters to control communication precisely.

Contracts govern the following types of endpoint group communications:

- Between ACI fabric application EPGs ($fvAEPg$), both intra-tenant and inter-tenant



Note In the case of a shared service mode, a contract is required for inter-tenant communication. A contract is used to specify static routes across VRFs, even though the tenant VRF does not enforce a policy.

- Between ACI fabric application EPGs and Layer 2 external outside network instance EPGs ($l2extInstP$)
- Between ACI fabric application EPGs and Layer 3 external outside network instance EPGs ($l3extInstP$)
- Between ACI fabric out-of-band ($mgmtOoB$) or in-band ($mgmtInB$) management EPGs

Contracts govern the communication between EPGs that are labeled providers, consumers, or both. EPG providers expose contracts with which a would-be consumer EPG must comply. The relationship between an EPG and a contract can be either a provider or consumer. When an EPG provides a contract, communication with that EPG can be initiated from other EPGs as long as the communication complies with the provided contract. When an EPG consumes a contract, the endpoints in the consuming EPG may initiate communication with any endpoint in an EPG that is providing that contract.



Note An EPG can both provide and consume the same contract. An EPG can also provide and consume multiple contracts simultaneously.

Configuring a Contract Using the REST API

SUMMARY STEPS

1. Configure a contract using an XML POST request similar to the following example:

DETAILED STEPS

Configure a contract using an XML POST request similar to the following example:

Example:

```
<vzBrCP name="webCtrct">
  <vzSubj name="http" revFltPorts="true" provmatchT="All">
    <vzRsSubjFiltAtt tnVzFilterName="Http"/>
    <vzRsSubjGraphAtt graphName="G1" termNodeName="TProv"/>
    <vzProvSubjLbl name="openProv"/>
    <vzConsSubjLbl name="openCons"/>
  </vzSubj>
  <vzSubj name="https" revFltPorts="true" provmatchT="All">
    <vzProvSubjLbl name="secureProv"/>
    <vzConsSubjLbl name="secureCons"/>
    <vzRsSubjFiltAtt tnVzFilterName="Https"/>
    <vzRsOutTermGraphAtt graphName="G2" termNodeName="TProv"/>
  </vzSubj>
</vzBrCP>
```

Configuring a Taboo Contract Using the REST API

Before you begin

The following objects must be created:

- The tenant that will be associated with this **Taboo Contract**
- An application profile for the tenant
- At least one EPG for the tenant

To create a taboo contract with the REST API, use XML such as in the following example:

Example:

```
<vzTaboo ownerTag="" ownerKey="" name="VRF64_Taboo_Contract"
dn="uni/tn-Tenant64/taboo-VRF64_Taboo_Contract" descr=""><vzTSubj
name="EPG_subject" descr=""><vzRsDenyRule tnVzFilterName="default"
directives="log"/>
</vzTSubj>
</vzTaboo>
```

Contract and Subject Exceptions

Configuring Contract or Subject Exceptions for Contracts

In Cisco APIC Release 3.2(1), contracts between EPGs are enhanced to enable denying a subset of contract providers or consumers from participating in the contract. Inter-EPG contracts and Intra-EPG contracts are supported with this feature.

You can enable a provider EPG to communicate with all consumer EPGs except those that match criteria configured in a subject or contract exception. For example, if you want to enable an EPG to provide services to all EPGs for a tenant, except a subset, you can enable those EPGs to be excluded. To configure this, you create an exception in the contract or one of the subjects in the contract. The subset is then denied access to providing or consuming the contract.

Labels, counters, and permit and deny logs are supported with contracts and subject exceptions.

To apply an exception to all subjects in a contract, add the exception to the contract. To apply an exception only to a single subject in the contract, add the exception to the subject.

When adding filters to subjects, you can set the action of the filter (to permit or deny objects that match the filter criteria). Also for **Deny** filters, you can set the priority of the filter. **Permit** filters always have the default priority. Marking the subject-to-filter relation to deny automatically applies to each pair of EPGs where there is a match for the subject. Contracts and subjects can include multiple subject-to-filter relationships that can be independently set to permit or deny the objects that match the filters.

Exception Types

Contract and subject exceptions can be based on the following types and include regular expressions, such as the * wildcard:

Exception criteria exclude these objects as defined in the Consumer Regex and Provider Regex fields	Example	Description
Tenant	<pre><vzException consRegex= "common" field= "Tenant" name= "excep03" provRegex= "t1" /></pre>	This example, excludes EPGs using the <code>common</code> tenant from consuming contracts provided by the <code>t1</code> tenant.
VRF	<pre><vzException consRegex= "ctx1" field= "Ctx" name= "excep05" provRegex= "ctx1" /></pre>	This example excludes members of <code>ctx1</code> from consuming the services provided by the same VRF.
EPG	<pre><vzException consRegex= "EPgPa.*" field= "EPg" name= "excep03" provRegex= "EPg03" /></pre>	The example assumes that multiple EPGs exist, with names starting with <code>EPGPa</code> , and they should all be denied as consumers for the contract provided by <code>EPg03</code> .
Dn	<pre><vzException consRegex= "uni/tn-t36/ap-customer/epg-epg193" field= "Dn" name="excep04" provRegex= "uni/tn-t36/ap-customer/epg-epg200" /></pre>	This example excludes <code>epg193</code> from consuming the contract provided by <code>epg200</code> .

Exception criteria exclude these objects as defined in the Consumer Regex and Provider Regex fields	Example	Description
Tag	<pre><vzException consRegex= "red" field= "Tag" name= "excep01" provRegex= "green" /></pre>	The example excludes objects marked with the <code>red</code> tag from consuming and those marked with the <code>green</code> tag from participating in the contract.

Configure a Contract or Subject Exception Using the REST API

In this task, you configure a contract that will allow most of the EPGs to communicate, but deny access to a subset of them. Multiple exceptions can be added to a contract or a subject.

Before you begin

Configure the tenant, VRF, application profile, and EPGs to provide and consume the contract.

Step 1 Create a filter by sending a post with XML, such as the following example:

Example:

```
<vzFilter name='http-filter'>
  <vzEntry name='http-e' etherT='ip' prot='tcp' />
  <vzEntry name='https-e' etherT='ip' prot='tcp' />
</vzFilter>
```

Step 2 Create a contract that excludes `EPg01` from consuming the subject and `EPg03` from providing it, by sending a post with XML, such as the following example:

The `vzException` MO can be contained by the `vzBrCP` or `vzSubj` MOs.

Example:

```
<vzBrCP name="httpCtrct" scope="context">
  <vzSubj name="subj1"
    <vzException consRegex="EPg01" field="EPg" name="excep01" provRegex=EPg03"/>
  </vzSubj />
  <vzRsSubjFiltAtt tnVzFilterName="http-filter" Action="deny"/>
  <vzRsSubjFiltAtt tnVzFilterName="https-filter" Action="permit"/>
</vzSubj>
</vzBrCP>
```

Configuring EPG Contract Inheritance Using the REST API

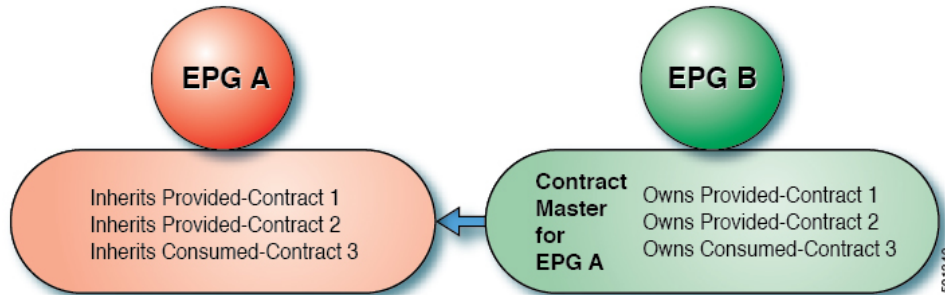
About Contract Inheritance

To streamline associating contracts to new EPGs, you can now enable an EPG to inherit all the (provided and consumed) contracts associated directly to another EPG in the same tenant. Contract inheritance can be configured for application, microsegmented, L2Out, and L3Out EPGs.

With Release 3.x, you can also configure contract inheritance for Inter-EPG contracts, both provided and consumed. Inter-EPG contracts are supported on Cisco Nexus 9000 Series switches with EX or FX at the end of their model name or later models.

You can enable an EPG to inherit all the contracts associated directly to another EPG, using the APIC GUI, NX-OS style CLI, and the REST API.

Figure 17: Contract Inheritance



In the diagram above, EPG A is configured to inherit Provided-Contract 1 and 2 and Consumed-Contract 3 from EPG B (contract master for EPG A).

Use the following guidelines when configuring contract inheritance:

- Contract inheritance can be configured for application, microsegmented (uSeg), external L2Out EPGs, and external L3Out EPGs. The relationships must be between EPGs of the same type.
- Both provided and consumed contracts are inherited from the contract master when the relationship is established.
- Contract masters and the EPGs inheriting contracts must be within the same tenant.
- Changes to the masters' contracts are propagated to all the inheritors. If a new contract is added to the master, it is also added to the inheritors.
- An EPG can inherit contracts from multiple contract masters.
- Contract inheritance is only supported to a single level (cannot be chained) and a contract master cannot inherit contracts.
- Labels with contract inheritance is supported. When EPG A inherits a contract from EPG B, if different subject labels are configured under EPG A and EPG B, APIC uses the label configured under EPG B for the contract inherited from EPG B. APIC uses the label configured under EPG A for the contract where EPG A is directly involved.
- Whether an EPG is directly associated to a contract or inherits a contract, it consumes entries in TCAM. So contract scale guidelines still apply. For more information, see the *Verified Scalability Guide* for your release.
- vzAny security contracts and taboo contracts are not supported.

For information about configuring Contract Inheritance and viewing inherited and standalone contracts, see *Cisco APIC Basic Configuration Guide*.

Configuring Application EPG Contract Inheritance Using the REST API

Before you begin

Configure the tenant and application profile to be used by the EPGs.

Configure the application EPG, to serve as the **EPG Contract Master**.

Configure the contracts to be shared, and associate them to the contract master.

To configure contract inheritance using the REST API, send a post with XML such as the following XML and JSON examples, with a URL directed to the EPG that will inherit the contracts:

Example:

XML Example

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- /api/node/mo/uni/tn-coke/ap-AP/epg-EPg_B.xml -->
<polUni>

  <fvEPg>

    <fvRsSecInherited tDn="uni/tn-coke/ap-AP/epg-EPg_B"/>
  </fvEPg>
</polUni>
```

JSON Example

```
https://192.168.200.10/api/node/mo/uni/tn-coke/ap-AP/epg-EPg_B.json
fvAEPg":{"attributes":{"dn":"uni/tn-coke/ap-AP/epg-EPg_B","name":"EPg_C",
"rn":"epg-EPg_C",
"status":"created"},
"children":[{"fvRsBd":{"attributes":{"tnFvBDName":"default",
"status":"created,modified"},
"children":[]}},{"fvRsSecInherited":{"attributes":{"tDn":"uni/tn-coke/ap-AP/epg-EPg_B",
"status":"created"},
"children":[]}}]}
```

Configuring uSeg EPG Contract Inheritance Using the REST API

Before you begin

Configure the tenant and application profile to be used by the EPGs.

Configure the application EPG, to serve as the **EPG Contract Master**.

Configure the contracts to be shared, and associate them to the contract master.

To configure uSeg contract inheritance using the REST API, send a post with XML such as the following example:

Example:

```

<polUni>
  <fvTenant name="Tn1" >
    <fvAEPg descr="" dn="uni/tn-Tn1/ap-AP1/epg-uSeg1_301_120" fwdCtrl="" isAttrBasedEPg="yes"
    matchT="AtleastOne" name="uSeg1_301_120" pcEnfPref="unenforced" prefGrMemb="exclude" prio="unspecified">

      <fvRsSecInherited tDn="uni/tn-Tn1/ap-AP1/epg-uSeg1_301_100" />
      <fvRsSecInherited tDn="uni/tn-Tn1/ap-AP1/epg-uSeg1_301_110" />
      <fvRsSecInherited tDn="uni/tn-Tn1/ap-AP1/epg-uSeg1_301_50" />
      <fvRsSecInherited tDn="uni/tn-Tn1/ap-AP1/epg-uSeg1_301_60" />
      <fvRsSecInherited tDn="uni/tn-Tn1/ap-AP1/epg-uSeg1_301_30" />
      <fvRsSecInherited tDn="uni/tn-Tn1/ap-AP1/epg-uSeg1_301_10" />
      <fvRsSecInherited tDn="uni/tn-Tn1/ap-AP1/epg-uSeg1_301_40" />
      <fvRsSecInherited tDn="uni/tn-Tn1/ap-AP1/epg-uSeg1_301_70" />
      <fvRsSecInherited tDn="uni/tn-Tn1/ap-AP1/epg-uSeg1_301_90" />
      <fvRsSecInherited tDn="uni/tn-Tn1/ap-AP1/epg-uSeg1_301_20" />
      <fvRsSecInherited tDn="uni/tn-Tn1/ap-AP1/epg-uSeg1_301_80" />
      <fvRsNodeAtt descr="" encap="unknown" instrImedcy="immediate" mode="regular"
    tDn="topology/pod-1/node-108" />
      <fvRsNodeAtt descr="" encap="unknown" instrImedcy="immediate" mode="regular"
    tDn="topology/pod-1/node-109" />
      <fvRsDomAtt classPref="encap" delimiter="" encap="vlan-301" encapMode="auto"
    instrImedcy="immediate" netflowPref="disabled" primaryEncap="unknown" resImedcy="immediate"
    tDn="uni/phys-PhysDom1" />
      <fvRsCustQosPol tnQosCustomPolName="" />
      <fvRsBd tnFvBDName="T1BD21" />
      <fvCrtrn descr="" match="any" name="default" nameAlias="" ownerKey="" ownerTag="" prec="0">

        <fvIpAttr descr="" ip="192.14.1.120" name="0" nameAlias="" ownerKey="" ownerTag=""
    usefvSubnet="no" />
      </fvCrtrn>
    </fvAEPg>
  </fvTenant>
</polUni>

```

What to do next**Configuring L2Out EPG Contract Inheritance Using the REST API****Before you begin**

Configure the tenant and application profile to be used by the EPGs.

Configure the L2Out EPG, to serve as the **L2Out Contract Master**.

Configure the contracts to be shared, and associate them to the contract master.

To configure L2Out EPG contract inheritance using the REST API, send a post with XML such as the following example:

Example:

```

<polUni>
  <fvTenant name="Tn1" >
    <l2extOut name="l2out1">
      <l2extRsEBd encap="vlan-51" tnFvBDName="T1BD1" />
      <l2extRsL2DomAtt tDn="uni/l2dom-l2Dom1" />
      <l2extLNodeP name="default" >
        <l2extLIfP name="default" >

```

```

        <l2extRsPathL2OutAtt tDn="topology/pod-1/protopaths-108-109/pathep-[VPC83]" />
    </l2extLIIfP>
</l2extLNodeP>
<l2extInstP matchT="AtleastOne" name="l2Ext1">
    <fvSubnet ctrl="nd" ip="192.13.1.10/24" preferred="no" scope="public,shared" virtual="no"
/>
    <fvRsProv tnVzBrCPName="T1ctr_tcp" />
</l2extInstP>
</l2extOut>

<l2extOut name="l2out2">
    <l2extRsEBd encap="vlan-53" tnFvBDName="T1BD3" />
    <l2extRsL2DomAtt tDn="uni/l2dom-l2Dom1" />
    <l2extLNodeP name="default" >
        <l2extLIIfP name="default" >
            <l2extRsPathL2OutAtt tDn="topology/pod-1/protopaths-108-109/pathep-[VPC84]" />
        </l2extLIIfP>
    </l2extLNodeP>
    <l2extInstP matchT="AtleastOne" name="l2Ext3" prefGrMemb="exclude">
        <fvSubnet ctrl="nd" ip="192.13.2.10/24" preferred="no" scope="public,shared" virtual="no"
/>
        <fvRsSecInherited tDn="uni/tn-Tn1/l2out-l2out1/instP-l2Ext1" />
    </l2extInstP>
</l2extOut>

</fvTenant>
</polUni>

```

Configuring L3Out EPG Contract Inheritance Using the REST API

Before you begin

Configure the tenant and application profile to be used by the EPGs.

Configure the L3Out EPG, to serve as the **L3Out Contract Master**.

Configure the contracts to be shared, and associate them to the contract master.

To configure L3Out EPG contract inheritance using the REST API, send a post with XML such as the following example:

Example:

```

<polUni>
    <fvTenant name="Tn6" >

        <!-- L3out creation -->
        <ospfIfPol deadIntvl="40" helloIntvl="10" name="ospf1" pfxSuppress="inherit" prio="1"
reemitIntvl="5" xmitDelay="1" />
        <l3extOut enforceRtctrl="export" name="T6L3out821">
            <ospfExtP areaCost="1" areaCtrl="redistribute,summary" areaId="0.0.0.1" areaType="regular"
/>
        />

        <l3extRsL3DomAtt tDn="uni/l3dom-L3Dom1" />
        <l3extRsEctx tnFvCtxName="T6ctx21" />
        <l3extLNodeP name="l3out_vpc82_prof" >
            <l3extRsNodeL3OutAtt rtrId="1.1.1.8" rtrIdLoopBack="yes" tDn="topology/pod-1/node-108">
                <l3extInfraNodeP fabricExtCtrlPeering="no" />
            </l3extRsNodeL3OutAtt>
            <l3extRsNodeL3OutAtt rtrId="1.1.1.9" rtrIdLoopBack="yes" tDn="topology/pod-1/node-109">
                <l3extInfraNodeP fabricExtCtrlPeering="no" />
            </l3extRsNodeL3OutAtt>
        </l3extLNodeP>
    </fvTenant>
</polUni>

```

```

    </l3extRsNodeL3OutAtt>
    <l3extLIIfP name="ospf1" >
      <ospfIfP authKeyId="1" authType="none" >
        <ospfRsIfPol tnOspfIfPolName="ospf1" />
      </ospfIfP>
      <l3extRsPathL3OutAtt encap="vlan-551" ifInstT="ext-svi" mode="regular" mtu="1500"
tDn="topology/pod-1/protopaths-108-109/pathep-[VPC82]" >
        <l3extMember addr="192.16.51.1/24" llAddr="0.0.0.0" side="B" />
        <l3extMember addr="192.16.51.2/24" llAddr="0.0.0.0" side="A" />
      </l3extRsPathL3OutAtt>
      <l3extRsNdIfPol tnNdIfPolName="" />
    </l3extLIIfP>
  </l3extLNodeP>

  <l3extInstP matchT="AtleastOne" name="T6l3Ext821">
    <fvRsProv tnVzBrCPName="T6ctr_UDP_TCP2" />
    <fvRsCons tnVzBrCPName="T6ctr_UDP_TCP1" />
    <l3extSubnet ip="192.16.51.0/24" scope="import-security,shared-rtctrl,shared-security" />

    <l3extSubnet ip="192.16.61.0/24" scope="import-security,shared-rtctrl,shared-security"
/>
    <vzConsSubjLbl name="tcp" tag="green" />
    <vzProvSubjLbl name="tcp" tag="green" />
  </l3extInstP>

  <l3extInstP matchT="AtleastOne" name="T6l3Ext823">
    <fvRsSecInherited tDn="uni/tn-Tn6/out-T6L3out821/instP-T6l3Ext821" />
    <l3extSubnet ip="192.16.63.0/24" scope="import-security,shared-rtctrl,shared-security"
/>
  </l3extInstP>
</l3extOut>

</fvTenant>
</polUni>

```

Contract Preferred Groups

About Contract Preferred Groups

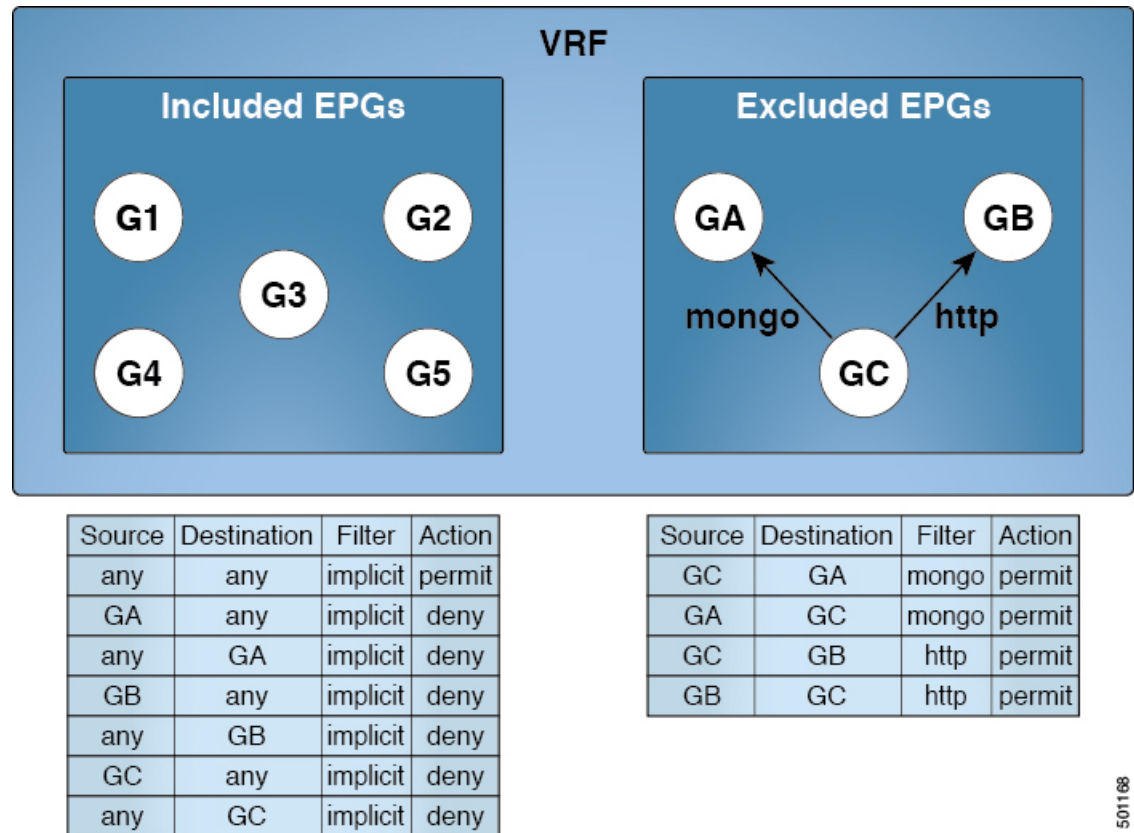
There are two types of policy enforcements available for EPGs in a VRF with a contract preferred group configured:

- **Included EPGs:** EPGs can freely communicate with each other without contracts, if they have membership in a contract preferred group. This is based on the source-any-destination-any-permit default rule.
- **Excluded EPGs:** EPGs that are not members of preferred groups require contracts to communicate with each other. Otherwise, the default source-any-destination-any-deny rule applies.

The contract preferred group feature enables greater control of communication between EPGs in a VRF. If most of the EPGs in the VRF should have open communication, but a few should only have limited communication with the other EPGs, you can configure a combination of a contract preferred group and contracts with filters to control inter-EPG communication precisely.

EPGs that are excluded from the preferred group can only communicate with other EPGs if there is a contract in place to override the source-any-destination-any-deny default rule.

Figure 18: Contract Preferred Group Overview



501168

Service Graph Support

As of APIC release 4.0(1), EPGs created by service graphs can be included in contract preferred groups. A new policy (Service EPG Policy) is available for defining the preferred group membership type (include or exclude). Once configured, it can be applied through the device selection policy or through the application of a service graph template.

Also, shadow EPGs can now be configured to be included or excluded in preferred groups.

Limitations

The following limitations apply to contract preferred groups:

- In topologies where an L3Out and application EPG are configured in a Contract Preferred Group, and the EPG is deployed only on a VPC, you may find that only one leaf switch in the VPC has the prefix entry for the L3Out. In this situation, the other leaf switch in the VPC does not have the entry, and therefore drops the traffic.

To workaroud this issue, you can do one of the following:

- Disable and reenale the contract group in the VRF
- Delete and recreate the prefix entries for the L3Out EPG

- Also, where the provider or consumer EPG in a service graph contract is included in a contract group, the shadow EPG can not be excluded from the contract group. The shadow EPG will be permitted in the contract group, but it does not trigger contract group policy deployment on the node where the shadow EPG is deployed. To download the contract group policy to the node, you deploy a dummy EPG within the contract group .
- Due to CSCvm63145, an EPG in a Contract Preferred Group can consume a shared service contract, but cannot be a provider for a shared service contract with an L3Out EPG as consumer.

Configuring Contract Preferred Groups Using the REST API

The following example creates a contract preferred group in `vrf64`, and creates three EPGs in the VRF:

- `epg-ldap`—Included in the preferred group
- `mail`—Included in the preferred group
- `radius`—Excluded from the preferred group

Before you begin

Create the tenants, VRFs, and the EPGs in the VRF.

Create a contract preferred group by sending a post, with XML such as the example:

Example:

```
<polUni>
  <fvTenant name="tenant64">
    <fvCtx name="vrf64"> <vzAny prefGrMemb="enabled"/> </fvCtx>
    <fvBD name="bd64"> <fvRsCtx tnFvCtxName="vrf64"/> </fvBD>
    <fvAp name="app-ldap">
      <fvAEPg name="epg-ldap" prefGrMemb="include">
        <fvRsBd tnFvBDName="bd64"/>
        <fvRsPathAtt tDn="topology/pod-1/paths-101/pathep-[eth1/3]" encap="vlan-113"
instrImedcy="immediate"/>
      </fvAEPg>
      <fvAEPg name="mail" prefGrMemb="include">
        <fvRsBd tnFvBDName="bd64"/>
        <fvRsPathAtt tDn="topology/pod-1/paths-101/pathep-[eth1/4]" encap="vlan-114"
instrImedcy="immediate"/>
      </fvAEPg>
      <fvAEPg name="radius" prefGrMemb="exclude">
        <fvRsBd tnFvBDName="bd64"/>
        <fvRsPathAtt tDn="topology/pod-1/paths-101/pathep-[eth1/5]" encap="vlan-115"
instrImedcy="immediate"/>
      </fvAEPg>
    </fvAp>
  </fvTenant>
</polUni>
```

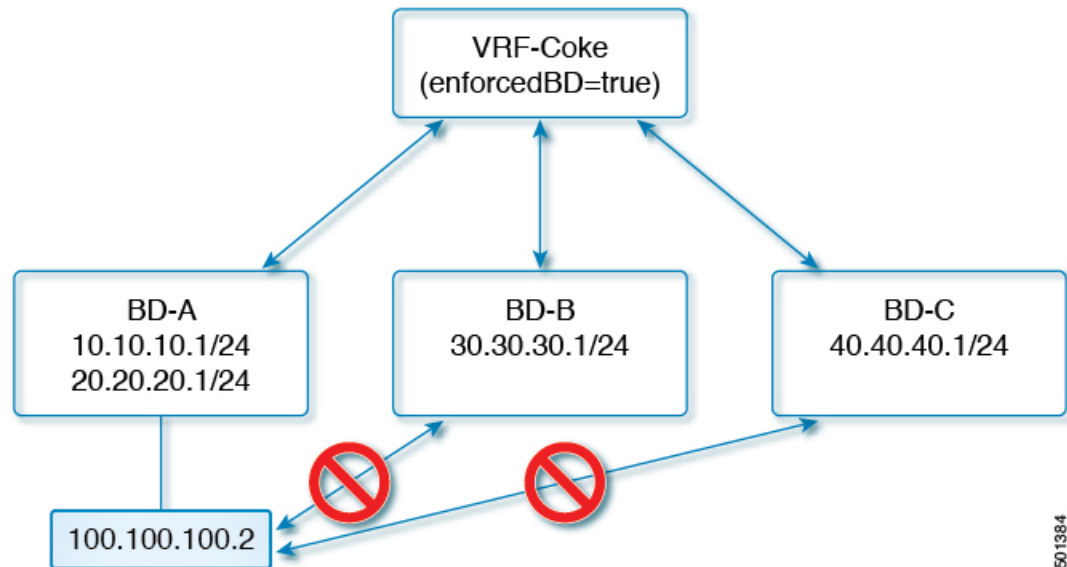
What to do next

Create a contract governing the communication of the `radius` EPG with other EPGs.

Configuring an Enforced Bridge Domain

An enforced bridge domain configuration entails creating an endpoint in a subject endpoint group (EPG) that can only ping subnet gateways within the associated bridge domain. With this configuration, you can then create a global exception list of IP addresses that can ping any subnet gateway.

Figure 19: Enforced Bridge Domain



501384



Note

- The exception IP addresses can ping all of the bridge domain gateways across all of your VRF instances.
- A loopback interface configured for an L3Out does not enforce reachability to the IP address that is configured for the subject loopback interface.
- When an eBGP peer IP address exists in a different subnet than the subnet of the L3Out interface, you must add the peer subnet to the allowed exception subnets. Otherwise, eBGP traffic is blocked because the source IP address exists in a different subnet than the L3Out interface subnet.
- For a BGP prefixed-based peer, you must add the peer subnet to the list of allowed exception subnets. For example, if 20.1.1.0/24 is configured as BGP prefixed-based peer, you must add 20.1.1.0/24 to the list of allowed exception subnets.
- An enforced bridge domain is not supported with the Management tenant, regardless if the VRF instances are in-band or out-of-band, and any rules to control the traffic to these VRF instances should be configured using regular contracts.

Configuring an Enforced Bridge Domain Using the REST API

SUMMARY STEPS

1. Create a tenant.
2. Create a VRF and bridge domain.

DETAILED STEPS

	Command or Action	Purpose
Step 1	<p>Create a tenant.</p> <p>Example:</p> <pre>POST https://apic-ip-address/api/mo/uni.xml <fvTenant name="ExampleCorp"/></pre>	When the POST succeeds, you see the object that you created in the output.
Step 2	<p>Create a VRF and bridge domain.</p> <p>Example:</p> <p>URL for POST: https://apic-ip-address/api/mo/uni/tn-ExampleCorp.xml</p> <pre><fvTenant name="ExampleCorp"> <fvCtx name="pvnl"/> <fvBD name="bd1"> <fvRsCtx tnFvCtxName="pvnl" bdEnforceEnable="yes"/> <fvSubnet ip="10.10.100.1/24"/> </fvBD> </fvTenant></pre> <p>For adding an exception IP, use the following post: https://apic-ip-address/api/node/mo/uni/infra.xml</p> <pre><bdEnforceExceptionCont> <bdEnforceExceptIp ip="11.0.1.0/24"/> </bdEnforceExceptionCont></pre> <p>Note If you have a public subnet when you configure the routed outside, you must associate the bridge domain with the outside configuration.</p>	<p>Note The Gateway Address can be an IPv4 or an IPv6 address. For more about details IPv6 gateway address, see the related KB article, <i>KB: Creating a Tenant, VRF, and Bridge Domain with IPv6 Neighbor Discovery</i>.</p>



CHAPTER 13

Provisioning Core Services

- [DHCP, on page 201](#)
- [DNS, on page 205](#)
- [NTP, on page 207](#)
- [Tetration, on page 208](#)
- [NetFlow, on page 209](#)
- [DOM Statistics, on page 214](#)
- [Syslog, on page 215](#)
- [Data Plane Policing, on page 218](#)
- [Traffic Storm Control, on page 220](#)
- [Rogue Endpoint Control, on page 222](#)

DHCP

Configuring a DHCP Relay Policy

A DHCP relay policy may be used when the DHCP client and server are in different subnets. If the client is on an ESX hypervisor with a deployed vShield Domain profile, then the use of a DHCP relay policy configuration is mandatory.

When a vShield controller deploys a Virtual Extensible Local Area Network (VXLAN), the hypervisor hosts create a kernel (vmkN, virtual tunnel end-point [VTEP]) interface. These interfaces need an IP address in the infrastructure tenant that uses DHCP. Therefore, you must configure a DHCP relay policy so that the Cisco Application Policy Infrastructure Controller (APIC) can act as the DHCP server and provide these IP addresses.

When a Cisco Application Centric Infrastructure (ACI) fabric acts as a DHCP relay, it inserts the DHCP Option 82 (the DHCP Relay Agent Information Option) in DHCP requests that it proxies on behalf of clients. If a response (DHCP offer) comes back from a DHCP server without Option 82, it is silently dropped by the fabric. Therefore, when the Cisco ACI fabric acts as a DHCP relay, DHCP servers providing IP addresses to compute nodes attached to the Cisco ACI fabric must support Option 82.

Configuring a DHCP Server Policy for the APIC Infrastructure Using the REST API

- This task is a prerequisite for users who want to create a vShield domain profile.

- The port and the encapsulation used by the application endpoint group must belong to a physical or VM Manager (VMM) domain. If no such association with a domain is established, the Cisco Application Policy Infrastructure Controller (APIC) continues to deploy the EPG but raises a fault.
- Cisco APIC supports DHCP relay for both IPv4 and IPv6 tenant subnets. DHCP server addresses can be IPv4 or IPv6. DHCPv6 relay will occur only if IPv6 is enabled on the fabric interface and one or more DHCPv6 relay servers are configured.

Before you begin

Make sure that Layer 2 or Layer 3 management connectivity is configured.

Configure the Cisco APIC as the DHCP server policy for the infrastructure tenant.

Note This relay policy will be pushed to all the leaf ports that are connected hypervisors using the attach entity profile configuration. For details about configuring with attach entity profile, see the examples related to creating VMM domain profiles.

Example:

DHCP Relay Policy for EPG

```
<!-- api/policymgr/mo/.xml -->
<polUni>

POST https://apic-ip-address/api/mo/uni.xml

<fvTenant name="infra">

  <dhcpRelayP name="DhcpRelayP" owner="tenant">
    <dhcpRsProv tDn="uni/tn-infra/ap-access/epg-default" addr="10.0.0.1" />
  </dhcpRelayP>

  <fvBD name="default">
    <dhcpLbl name="DhcpRelayP" owner="tenant"/>
  </fvBD>

</fvTenant>
</polUni>
```

Example:

DHCP Relay Policy for Layer 3 Outside

Note You must specify DHCP Relay label under **l3extLifP** with an appropriate name and owner.

```
<polUni>
  <fvTenant name="dhcpTn">
    <l3extOut name="Out1" >
      <l3extLNodeP name="NodeP" >
        <l3extLIfP name="Intf1">
          <dhcpLbl name="DhcpRelayPol" owner="tenant" />
        </l3extLIfP>
      </l3extLNodeP>
    </l3extOut>
  </fvTenant>
</polUni>
```

POST <https://apic-ip-address/api/mo/uni.xml>

Layer 2 and Layer 3 DHCP Relay Sample Policies

This sample policy provides an example of a consumer tenant L3extOut DHCP relay configuration:

```
<polUni>
  <!-- Consumer Tenant 2 -->
  <fvTenant
    dn="uni/tn-tenant1"
    name="tenant1">
    <fvCtx name="dhcp"/>

    <!-- DHCP client bridge domain -->
    <fvBD name="cons2">
      <fvRsBDToOut tnL3extOutName='L3OUT'/>
      <fvRsCtx tnFvCtxName="dhcp" />
      <fvSubnet ip="20.20.20.1/24"/>
      <dhcpLbl name="DhcpRelayP" owner="tenant"/>
    </fvBD>
    <!-- L3Out EPG DHCP -->
    <l3extOut name="L3OUT">
      <l3extRsEctx tnFvCtxName="dhcp"/>
      <l3extInstP name="l3extInstP-1">
        <!-- Allowed routes to L3out to send traffic -->
        <l3extSubnet ip="100.100.100.0/24" />
      </l3extInstP>
      <l3extLNodeP name="l3extLNodeP-pc">
        <!-- VRF External loopback interface on node -->
        <l3extRsNodeL3OutAtt
          tDn="topology/pod-1/node-1018"
          rtrId="10.10.10.1" />
        <l3extLIIfP name='l3extLIIfP-pc'>
          <l3extRsPathL3OutAtt
            tDn="topology/pod-1/paths-1018/pathep-[eth1/7]"
            encap='vlan-900'
            ifInstT='sub-interface'
            addr="100.100.100.50/24"
            mtu="1500"/>
          </l3extLIIfP>
        </l3extLNodeP>
      </l3extOut>
    <!-- Static DHCP Client Configuration -->
    <fvAp name="cons2">
      <fvAEPg name="APP">
        <fvRsBd tnFvBDName="cons2"/>
        <fvRsDomAtt tDn="uni/phys-mininet"/>
        <fvRsPathAtt
          tDn="topology/pod-1/paths-1017/pathep-[eth1/3]"
          encap="vlan-1000"
          instrImedcy='immediate'
          mode='native' />
        </fvAEPg>
      </fvAp>
    <!-- DHCP Server Configuration -->
    <dhcpRelayP
      name="DhcpRelayP"
      owner="tenant"
      mode="visible">
```

```

        <dhcpRsProv
            tDn="uni/tn-tenant1/out-L3OUT/instP-l3extInstP-1"
            addr="100.100.100.1"/>
        </dhcpRelayP>
    </fvTenant>
</polUni>

```

This sample policy provides an example of a consumer tenant L2extOut DHCP relay configuration:

```

<fvTenant
    dn="uni/tn-dhcpl2Out"
    name="dhcpl2Out">
    <fvCtx name="dhcpl2Out"/>
        <!-- bridge domain -->

    <fvBD name="provBD">
        <fvRsCtx tnFvCtxName="dhcpl2Out" />
            <fvSubnet ip="100.100.100.50/24" scope="shared"/>
    </fvBD>

    <!-- Consumer bridge domain -->
    <fvBD name="cons2">
        <fvRsCtx tnFvCtxName="dhcpl2Out" />
            <fvSubnet ip="20.20.20.1/24"/>
            <dhcpLbl name="DhcpRelayP" owner="tenant"/>
    </fvBD>

    <vzFilter name='t0f0' >
    <vzEntry name='t0f0e9'></vzEntry>
    </vzFilter>

    <vzBrCP name="webCtrct" scope="global">
        <vzSubj name="app">
            <vzRsSubjFiltAtt tnVzFilterName="t0f0"/>
        </vzSubj>
    </vzBrCP>

    <l2extOut name="l2Out">
        <l2extLNodeP name='l2ext'>
        <l2extLIIfP name='l2LifP'>
            <l2extRsPathL2OutAtt tDn="topology/pod-1/paths-1018/pathep-[eth1/7]"/>
        </l2extLIIfP>
        </l2extLNodeP>
        <l2extInstP name='l2inst'>
            <fvRsProv tnVzBrCPName="webCtrct"/>
        </l2extInstP>
        <l2extRsEBd tnFvBDName="provBD" encap='vlan-900' />
    </l2extOut>

    <fvAp name="cons2">
        <fvAEPg name="APP">
            <fvRsBd tnFvBDName="cons2" />
                <fvRsDomAtt tDn="uni/phys-mininet" />
                <fvRsBd tnFvBDName="SolarBD2" />
                <fvRsPathAtt tDn="topology/pod-1/paths-1018/pathep-[eth1/48]"
encap="vlan-1000" instrImedcy='immediate' mode='native' />
            </fvAEPg>
        </fvAp>
        <dhcpRelayP name="DhcpRelayP" owner="tenant" mode="visible">
            <dhcpRsProv tDn="uni/tn-dhcpl2Out/l2out-l2Out/instP-l2inst" addr="100.100.100.1"/>
        </dhcpRelayP>
    </fvTenant>

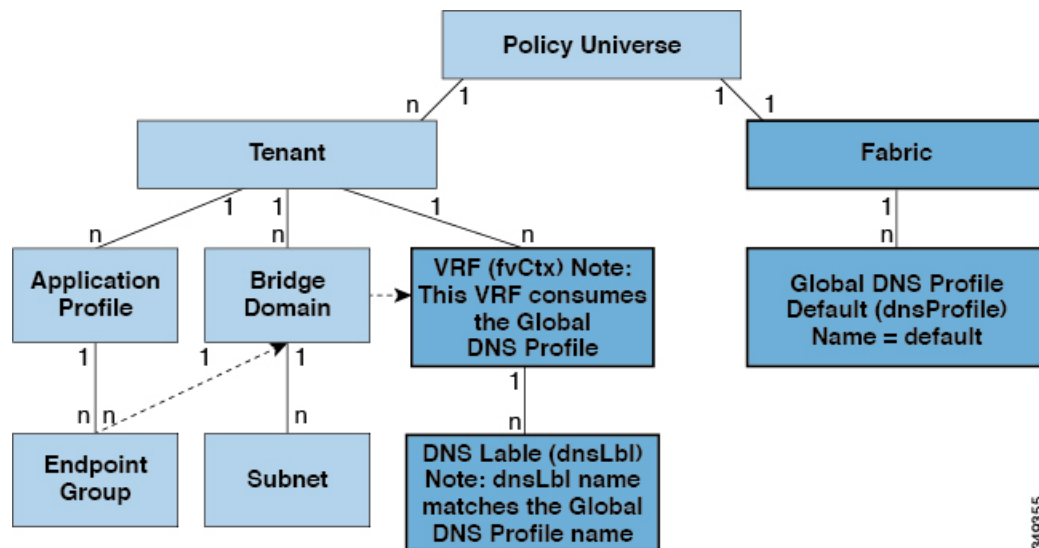
```

DNS

DNS

The ACI fabric DNS service is contained in the fabric managed object. The fabric global default DNS profile can be accessed throughout the fabric. The figure below shows the logical relationships of the DNS-managed objects within the fabric.

Figure 20: DNS



A VRF (context) must contain a `dnsLBl` object in order to use the global default DNS service. Label matching enables tenant VRFs to consume the global DNS provider. Because the name of the global DNS profile is “default,” the VRF label name is “default” (`dnsLBl name = default`).

Configuring a DNS Service Policy to Connect with DNS Providers Using the REST API

Before you begin

Make sure that Layer 2 or Layer 3 management connectivity is configured.

SUMMARY STEPS

1. Configure the DNS service policy.
2. Configure the DNS label under the out-of-band management tenant.

DETAILED STEPS

Step 1 Configure the DNS service policy.

Example:

```
POST URL :
https://apic-IP-address/api/node/mo/uni/fabric.xml

<dnsProfile name="default">

  <dnsProv addr="172.21.157.5" preferred="yes"/>
  <dnsProv addr="172.21.157.6"/>

  <dnsDomain name="cisco.com" isDefault="yes"/>

  <dnsRsProfileToEpg tDn="uni/tn-mgmt/mgmt-default/oob-default"/>

</dnsProfile>
```

Step 2 Configure the DNS label under the out-of-band management tenant.

Example:

```
POST URL: https://apic-IP-address/api/node/mo/uni/tn-mgmt/ctx-oob.xml
<dnsLbl name="default" tag="yellow-green"/>
```

DNS Policy Example

This sample policy creates a DNS profile and associates it with a tenant.

Create the DNS profile:

```
<!-- /api/policymgr/mo/.xml -->
<polUni>
<fabricInst>
<dnsProfile name="default">
  <dnsProv addr="172.21.157.5" preferred="yes"/>
  <dnsDomain name="insieme.local" isDefault="yes"/>
  <dnsRsProfileToEpg tDn="uni/tn-mgmt/mgmt-default/oob-default"/>
</dnsProfile>
</fabricInst>
</polUni>
```

Associate the profile with the tenant that will consume it:

```
<!-- /api/policymgr/mo/.xml -->
<polUni>
<fvTenant name='t1'>
  <fvCtx name='ctx0'>
    <dnsLbl name='default' />
  </fvCtx>
</fvTenant>
</polUni>
```


NTP

Time Synchronization and NTP

Within the Cisco Application Centric Infrastructure (ACI) fabric, time synchronization is a crucial capability upon which many of the monitoring, operational, and troubleshooting tasks depend. Clock synchronization is important for proper analysis of traffic flows as well as for correlating debug and fault time stamps across multiple fabric nodes.

An offset present on one or more devices can hamper the ability to properly diagnose and resolve many common operational issues. In addition, clock synchronization allows for the full utilization of the atomic counter capability that is built into the ACI upon which the application health scores depend. Nonexistent or improper configuration of time synchronization does not necessarily trigger a fault or a low health score. You should configure time synchronization before deploying a full fabric or applications so as to enable proper usage of these features. The most widely adapted method for synchronizing a device clock is to use Network Time Protocol (NTP).

Prior to configuring NTP, consider what management IP address scheme is in place within the ACI fabric. There are two options for configuring management of all ACI nodes and Application Policy Infrastructure Controllers (APICs), in-band management and/or out-of-band management. Depending upon which management option is chosen for the fabric, configuration of NTP will vary. Another consideration in deploying time synchronization is where the time source is located. The reliability of the source must be carefully considered when determining if you will use a private internal clock or an external public clock.

Configuring NTP Using the REST API



Note There is a risk of hostname resolution failure for hostname based NTP servers if the DNS server used is configured to be reachable over in-band or out-of-band connectivity. If you use a hostname, ensure that the DNS service policy to connect with the DNS providers is configured. Also ensure that the appropriate DNS label is configured for the in-band or out-of-band VRF instances of the management EPG that you chose when you configured the DNS profile policy.

Step 1 Configure NTP.

Example:

POST url: <https://APIC-IP/api/node/mo/uni/fabric/time-test.xml>

```
<imdata totalCount="1">
  <datetimePol adminSt="enabled" authSt="disabled" descr="" dn="uni/fabric/time-CiscoNTPPol"
name="CiscoNTPPol" ownerKey="" ownerTag="">
    <datetimeNtpProv descr="" keyId="0" maxPoll="6" minPoll="4" name="10.10.10.11" preferred="yes">
      <datetimeRsNtpProvToEpg tDn="uni/tn-mgmt/mgmt-pol-default/inb-default"/>
    </datetimeNtpProv>
  </datetimePol>
</imdata>
```

Step 2 Add the default Date Time Policy to the pod policy group.

Example:

```
POST url: https://APIC-IP/api/node/mo/uni/fabric/funcprof/podpgrp-calol/rsTimePol.xml

POST payload: <imdata totalCount="1">
<fabricRsTimePol tnDatetimePolName="CiscoNTFPol">
</fabricRsTimePol>
</imdata>
```

Step 3 Add the pod policy group to the default pod profile.

Example:

```
POST url: https://APIC-IP/api/node/mo/uni/fabric/podprof-default/pods-default-typ-ALL/rspodPGrp.xml

payload: <imdata totalCount="1">
<fabricRsPodPGrp tDn="uni/fabric/funcprof/podpgrp-calol" status="created">
</fabricRsPodPGrp>
</imdata>
```

Tetration

Overview

This article provides examples of how to configure Cisco Tetration when using the Cisco APIC. The following information applies when configuring Cisco Tetration.

- An inband management IP address must be configured on each leaf where the Cisco Tetration agent is active.
- Define an analytics policy and specify the destination IP address of the Cisco Tetration server.
- Create a switch profile and include the policy group created in the previous step.

Configuring Cisco Tetration Analytics Using the REST API

Step 1 Create the analytics policy.

Example:

```
<analyticsCluster name="tetration" >
<analyticsCfgSrv name="srv1" ip="10.30.30.7" >
</analyticsCfgSrv>
</analyticsCluster>
```

Step 2 Associate analytics with the policy group.

Example:

```
<fabricLeNodePGrp descr="" name="mypolicy6" ownerKey="" ownerTag="" rn="lenodepgrp-mypolicy6" status="">

  <fabricRsNodeCfgSrv rn="rsnodeProv" status=""
tDn="uni/fabric/analytics/cluster-tetration/cfgsrv-srv1" />
</fabricLeNodePGrp>
```

Step 3 Associate the policy group with the switch.

Example:

```
<fabricLeafP name="leafs" rn="leprof-leafs" status="" >
  <fabricLeafS name="sw" rn="leaves-sw-tyr-range" status="">
    <fabricRsLeNodePGrp rn="rsleNodePGrp" tDn="uni/fabric/funcprof/lenodepgrp-mypolicy6"/>
    <fabricNodeBlk name="switches" from_="101" to_="101" />
  </fabricLeafS>
</fabricLeafP>
```

NetFlow

About NetFlow

The NetFlow technology provides the metering base for a key set of applications, including network traffic accounting, usage-based network billing, network planning, as well as denial of services monitoring, network monitoring, outbound marketing, and data mining for both service providers and enterprise customers. Cisco provides a set of NetFlow applications to collect NetFlow export data, perform data volume reduction, perform post-processing, and provide end-user applications with easy access to NetFlow data. If you have enabled NetFlow monitoring of the traffic flowing through your datacenters, this feature enables you to perform the same level of monitoring of the traffic flowing through the Cisco Application Centric Infrastructure (Cisco ACI) fabric.

Instead of hardware directly exporting the records to a collector, the records are processed in the supervisor engine and are exported to standard NetFlow collectors in the required format.

For detailed information about configuring and using NetFlow, see *Cisco APIC and NetFlow*.

For information about configuring NetFlow with virtual machine networking, see the *Cisco ACI Virtualization Guide*.

NetFlow on EX Platform Switches

In addition to the generic support information, the following limitations apply to EX platform switches:

- NetFlow can be supported on a bridge domain; however, NetFlow cannot distinguish between bridged and routed packets. If you configure NetFlow on an interface VLAN (SVI) to capture only routed packets, NetFlow cannot limit collection to this type in EX switches.
- EX switches cannot provide an encapsulation VLAN in the flow record.
- EX switches do not have a MAC address packet classify feature, so the configuration engine flow record will contain only non-IP address flows (ARP is already treated as IP).
- EX switches do not support regularly-deployed and understood NetFlow sampling, such as packet-based sampling (M out of N).
- Having a type of service or source interface as part of the flow hash is not supported. Source interface information is collected in the record, but no type of service information is collected in EX switches.
- EX switches have fixed flow collection parameters.

- EX switches support only two flow records of each type. The exception is that four configuration engine flow records are supported.
- EX switches assign the following protocol numbers to identify the ARP and ND packets:
 - ARP Req 249
 - ARP Res 250
 - RARP Req 247
 - RARP Res 248
 - Nd Sol 249
 - Nd Adv 250

All other ARP and ND packets are set to 255.

Configuring a NetFlow Exporter Policy for VM Networking Using the REST API

The following example XML shows how to configure a NetFlow exporter policy for VM networking using the REST API:

```
<polUni>
  <infraInfra>
    <netflowVmmExporterPol name="vmExporter1" dstAddr="2.2.2.2" dstPort="1234"
srcAddr="4.4.4.4"/>
  </infraInfra>
</polUni>
```

Configuring NetFlow Infra Selectors Using REST API

You can use the REST API to configure NetFlow infra selectors. The infra selectors are used for attaching a Netflow monitor to a PHY, port channel, virtual port channel, fabric extender (FEX), or port channel fabric extender (FEXPC) interface.

The following example XML shows how to configure NetFlow infra selectors using the REST API:

```
<infraInfra>
  <!--Create Monitor Policy /-->
  <netflowMonitorPol name='monitor_policy1' descr='This is a monitor policy.'>
    <netflowRsMonitorToRecord tnNetflowRecordPolName='record_policy1' />
    <!-- A Max of 2 exporters allowed per Monitor Policy /-->
    <netflowRsMonitorToExporter tnNetflowExporterPolName='exporter_policy1' />
    <netflowRsMonitorToExporter tnNetflowExporterPolName='exporter_policy2' />
  </netflowMonitorPol>

  <!--Create Record Policy /-->
  <netflowRecordPol name='record_policy1' descr='This is a record policy.'
match='src-ipv4,src-port'/>

  <!--Create Exporter Policy /-->
  <netflowExporterPol name='exporter_policy1' dstAddr='10.10.1.1' srcAddr='10.10.1.10'
ver='v9' descr='This is an exporter policy.'>
    <!--Exporter can be behind app EPG or external L3 EPG (InstP) /-->
    <netflowRsExporterToEPG tDn='uni/tn-t1/ap-app1/epg-epg1'/>
  </netflowExporterPol>
</infraInfra>
```

```

        <!--This Ctx needs to be the same Ctx that EPG1's BD is part of /-->
        <netflowRsExporterToCtx tDn='uni/tn-t1/ctx-ctx1' />
    </netflowExporterPol>

    <!--Node-level Policy for collection Interval /-->
    <netflowNodePol name='node_policy1' collectIntvl='500' />

    <!-- Node Selectors - usual config /-->
    <infraNodeP name="infraNodeP-17" >
        <infraLeafS name="infraLeafS-17" type="range">
            <!-- NOTE: The nodes can also be fex nodes /-->
            <infraNodeBlk name="infraNodeBlk-17" from_"101" to_"101"/>
            <infraRsAccNodePGrp tDn='uni/infra/funcprof/accnodepgrp-nodePGrp1' />
        </infraLeafS>
        <infraRsAccPortP tDn="uni/infra/accportprof-infraAccPortP"/>
    </infraNodeP>

    <!-- Port Selectors - usual config /-->
    <infraAccPortP name="infraAccPortP" >
        <infraHPortS name="infraHPortS" type="range">
            <!-- NOTE: The interfaces can also be Port-channels, fex interfaces or fex PCs
 /-->
            <infraPortBlk name="infraPortBlk" fromCard="1" toCard="1" fromPort="8"
toPort="8"/>
            <infraRsAccBaseGrp tDn="uni/infra/funcprof/accportgrp-infraAccPortGrp"/>
        </infraHPortS>
    </infraAccPortP>

    <!-- Policy Groups - usual config /-->
    <infraFuncP>
        <!-- Node Policy Group - to setup Netflow Node Policy /-->
        <infraAccNodePGrp name='nodePGrp1' >
            <infraRsNetflowNodePol tnNetflowNodePolName='node_policy1' />
        </infraAccNodePGrp>

        <!-- Access Port Policy Group - to setup Netflow Monitor Policy /-->
        <infraAccPortGrp name="infraAccPortGrp" >
            <!--One Monitor Policy per address family (ipv4, ipv6, ce) /-->
            <infraRsNetflowMonitorPol tnNetflowMonitorPolName='monitor_policy1'
fltType='ipv4'/>
            <infraRsNetflowMonitorPol tnNetflowMonitorPolName='monitor_policy2'
fltType='ipv6'/>
            <infraRsNetflowMonitorPol tnNetflowMonitorPolName='monitor_policy2' fltType='ce'/>
        </infraAccPortGrp>
    </infraFuncP>
</infraInfra>

```

Configuring NetFlow Tenant Hierarchy Using REST API

You can use the REST API to configure the NetFlow tenant hierarchy. The tenant hierarchy is used for attaching a NetFlow monitor to a bridge domain, Layer 3 sub-interface, or Layer 3 switched virtual interface (SVI).

The following example XML shows how to configure the NetFlow tenant hierarchy using the REST API:

```

<?xml version="1.0" encoding="UTF-8"?>

<!-- api/policymgr/mo/.xml -->
<polUni>
    <fvTenant name="t1">

        <!--Create Monitor Policy /-->

```

```

<netflowMonitorPol name='monitor_policy1' descr='This is a monitor policy.'>
  <netflowRsMonitorToRecord tnNetflowRecordPolName='record_policy1' />
  <!-- A Max of 2 exporters allowed per Monitor Policy /-->
  <netflowRsMonitorToExporter tnNetflowExporterPolName='exporter_policy1' />
  <netflowRsMonitorToExporter tnNetflowExporterPolName='exporter_policy2' />
</netflowMonitorPol>
<!--Create Record Policy /-->
<netflowRecordPol name='record_policy1' descr='This is a record policy.'/>

<!--Create Exporter Policy /-->
<netflowExporterPol name='exporter_policy1' dstAddr='10.0.0.1' srcAddr='10.0.0.4'>

  <!--Exporter can be behind app EPG or external L3 EPG (InstP) /-->
  <netflowRsExporterToEPG tDn='uni/tn-t1/ap-app1/epg-epg2' />
  <!--netflowRsExporterToEPG tDn='uni/tn-t1/out-out1/instP-accountingInst' /-->
  <!--This Ctx needs to be the same Ctx that EPG2's BD is part of /-->
  <netflowRsExporterToCtx tDn='uni/tn-t1/ctx-ctx1' />
</netflowExporterPol>

<!--Create 2nd Exporter Policy /-->
<netflowExporterPol name='exporter_policy2' dstAddr='11.0.0.1' srcAddr='11.0.0.4'>
  <netflowRsExporterToEPG tDn='uni/tn-t1/ap-app1/epg-epg2' />
  <netflowRsExporterToCtx tDn='uni/tn-t1/ctx-ctx1' />
</netflowExporterPol>

<fvCtx name="ctx1" />

<fvBD name="bd1" unkMacUcastAct="proxy" >
  <fvSubnet descr="" ip="11.0.0.0/24"\>
  <fvRsCtx tnFvCtxName="ctx1" />

  <!--One Monitor Policy per address family (ipv4, ipv6, ce) /-->
  <fvRsBDToNetflowMonitorPol tnNetflowMonitorPolName='monitor_policy1'
fltType='ipv4' />
  <fvRsBDToNetflowMonitorPol tnNetflowMonitorPolName='monitor_policy2'
fltType='ipv6' />
  <fvRsBDToNetflowMonitorPol tnNetflowMonitorPolName='monitor_policy2'
fltType='ce' />
</fvBD>

<!--Create App EPG /-->
<fvAp name="app1">
  <fvAEPg name="epg2" >
    <fvRsBd tnFvBDName="bd1" />
    <fvRsPathAtt encap="vlan-20" instrImedcy="lazy" mode="regular"
tDn="topology/pod-1/paths-101/pathep-[eth1/20]" />
  </fvAEPg>
</fvAp>

<!--L3 Netflow Config for sub-intf and SVI /-->
<l3extOut name="out1">
  <l3extLNodeP name="lnodep1" >
    <l3extRsNodeL3OutAtt tDn="topology/pod-1/node-101" rtrId="1.2.3.4" />
    <l3extLIfP name='lifp1'>
      <!--One Monitor Policy per address family (ipv4, ipv6, ce) /-->
      <l3extRsLIfPToNetflowMonitorPol tnNetflowMonitorPolName='monitor_policy1'
fltType='ipv4' />
      <l3extRsLIfPToNetflowMonitorPol tnNetflowMonitorPolName='monitor_policy2'
fltType='ipv6' />
      <l3extRsLIfPToNetflowMonitorPol tnNetflowMonitorPolName='monitor_policy2'
fltType='ce' />

      <!--Sub-interface 1/40.40 on node 101 /-->
      <l3extRsPathL3OutAtt tDn="topology/pod-1/paths-101/pathep-[eth1/40]"

```

```

ifInstT='sub-interface' encap='vlan-40' />
    <!--SVI 50 attached to eth1/25 on node 101 /-->
    <l3extRsPathL3OutAtt tDn="topology/pod-1/paths-101/pathep-[eth1/25]"
ifInstT='external-svi' encap='vlan-50' />
    </l3extLIIfP>
    </l3extLNodeP>

    <!--External L3 EPG for Exporter behind external L3 Network /-->
    <l3extInstP name="accountingInst">
        <l3extSubnet ip="11.0.0.0/24" />
    </l3extInstP>
    <l3extRsEctx tnFvCtxName="ctx1"/>
    </l3extOut>
</fvTenant>
</polUni>

```

Consuming a NetFlow Exporter Policy Under a VMM Domain Using the REST API for VMware VDS

The following example XML shows how to consume a NetFlow exporter policy under a VMM domain using the REST API:

```

<polUni>
  <vmmProvP vendor="VMware">
    <vmmDomP name="mininet">
      <vmmVSwitchPolicyCont>
        <vmmRsVswitchExporterPol tDn="uni/infra/vmmexporterpol-vmExporter1"
activeFlowTimeOut="62" idleFlowTimeOut="16" samplingRate="1"/>
      </vmmVSwitchPolicyCont>
    </vmmDomP>
  </vmmProvP>
</polUni>

```

Configuring NetFlow or Tetration Analytics Priority Using REST API

You can specify whether to use the NetFlow or Cisco Tetration Analytics feature by setting the `FeatureSel` attribute of the `<fabricNodeControl>` element. The `FeatureSel` attribute can have one of the following values:

- `analytics`—Specifies Cisco Tetration Analytics. This is the default value.
- `netflow`—Specifies NetFlow.

The following example REST API post specifies for the switch "test1" to use the NetFlow feature:

```

http://192.168.10.1/api/node/mo/uni/fabric.xml
<fabricNodeControl name="test1" FeatureSel="netflow" />

```

DOM Statistics

About Digital Optical Monitoring

Real-time digital optical monitoring (DOM) data is collected from SFPs, SFP+, and XFPs periodically and compared with warning and alarm threshold table values. The DOM data collected are transceiver transmit bias current, transceiver transmit power, transceiver receive power, and transceiver power supply voltage.

Enabling Digital Optical Monitoring Using the REST API

Before you can view digital optical monitoring (DOM) statistics about a physical interface, enable DOM on the interface.

To enable DOM using the REST API:

Step 1 Create a fabric node control policy (fabricNodeControlPolicy) as in the following example:

```
<fabricNodeControl dn="uni/fabric/nodecontrol-testdom" name="testdom" control="1"
rn="nodecontrol-testdom" status="created" />
```

Step 2 Associate a fabric node control policy to a policy group as follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
<fabricLeNodePGrp dn="uni/fabric/funcprof/lenodepgrp-nodegrp2" name="nodegrp2"
rn="lenodepgrp-nodegrp2" status="created,modified" >

  <fabricRsMonInstFabricPol tnMonFabricPolName="default" status="created,modified" />
  <fabricRsNodeCtrl tnFabricNodeControlName="testdom" status="created,modified" />

</fabricLeNodePGrp>
```

Step 3 Associate a policy group to a switch (in the following example, the switch is 103) as follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
<fabricLeafP>
  <attributes>
    <dn>uni/fabric/leprof-leafSwitchProfile</dn>
    <name>leafSwitchProfile</name>
    <rn>leprof-leafSwitchProfile</rn>
    <status>created,modified</status>
  </attributes>
  <children>
    <fabricLeafS>
      <attributes>
        <dn>uni/fabric/leprof-leafSwitchProfile/leaves-test-typrange</dn>
        <type>range</type>
        <name>test</name>
        <rn>leaves-test-typrange</rn>
        <status>created,modified</status>
      </attributes>
      <children>
        <fabricNodeBlk>
          <attributes>
            <dn>uni/fabric/leprof-leafSwitchProfile/leaves-test-typrange/nodeblk-09533c1d228097da</dn>
```



```

    <from_>103</from_>
    <to_>103</to_>
    <name>09533c1d228097da</name>
    <rn>nodeblk-09533c1d228097da</rn>
    <status>created,modified</status>
  </attributes>
</fabricNodeBlk>
</children>
<children>
  <fabricRsLeNodePGrp>
    <attributes>
      <tDn>uni/fabric/funcprof/lenodepgrp-nodegrp2</tDn>
      <status>created</status>
    </attributes>
  </fabricRsLeNodePGrp>
</children>
</fabricLeafS>
</children>
</fabricLeafP>

```

Syslog

About Syslog

During operation, a fault or event in the Cisco Application Centric Infrastructure (ACI) system can trigger the sending of a system log (syslog) message to the console, to a local file, and to a logging server on another system. A system log message typically contains a subset of information about the fault or event. A system log message can also contain audit log and session log entries.



Note For a list of syslog messages that the APIC and the fabric nodes can generate, see http://www.cisco.com/c/en/us/td/docs/switches/datacenter/aci/apic/sw/1-x/syslog/guide/aci_syslog/ACI_SysMsg.html.

Many system log messages are specific to the action that a user is performing or the object that a user is configuring or administering. These messages can be the following:

- Informational messages, providing assistance and tips about the action being performed
- Warning messages, providing information about system errors related to an object, such as a user account or service profile, that the user is configuring or administering

In order to receive and monitor system log messages, you must specify a syslog destination, which can be the console, a local file, or one or more remote hosts running a syslog server. In addition, you can specify the minimum severity level of messages to be displayed on the console or captured by the file or host. The local file for receiving syslog messages is `/var/log/external/messages`.

A syslog source can be any object for which an object monitoring policy can be applied. You can specify the minimum severity level of messages to be sent, the items to be included in the syslog messages, and the syslog destination.

You can change the display format for the Syslogs to NX-OS style format.

Additional details about the faults or events that generate these system messages are described in the *Cisco APIC Faults, Events, and System Messages Management Guide*, and system log messages are listed in the *Cisco ACI System Messages Reference Guide*.



Note Not all system log messages indicate problems with your system. Some messages are purely informational, while others may help diagnose problems with communications lines, internal hardware, or the system software.

Configuring a Syslog Group and Destination Using the REST API

This procedure configures syslog data destinations for logging and evaluation. You can export syslog data to the console, to a local file, or to one or more syslog servers in a destination group. This example sends alerts to the console, information to a local file, and warnings to a remote syslog server.

To create a syslog group and destination using the REST API, send a post with XML such as the following example:

Example:

```
<syslogGroup name name="tenant64_SyslogDest" format="aci" dn="uni/fabric/slgroup-tenant64_SyslogDest">
  <syslogConsole name="" format="aci" severity="alerts" adminState="enabled"/>
  <syslogFile name="" format="aci" severity="information" adminState="enabled"/>
  <syslogProf name="syslog" adminState="enabled"/>
  <syslogRemoteDest name="Syslog_remoteDest" format="aci" severity="warnings"
    adminState="enabled" port="514" host="192.168.100.20" forwardingFacility="local7">
    <fileRsARemoteHostToEpg tDn="uni/tn-mgmt/mgmtp-default/oob-default"/>
  </syslogRemoteDest>
</syslogGroup>
```

Creating a Syslog Source Using the REST API

A syslog source can be any object for which an object monitoring policy can be applied.

Before you begin

Create a syslog monitoring destination group.

To create a syslog source, send a POST request with XML such as the following example:

Example:

```
<syslogSrc
  name="VRF64_SyslogSource" minSev="warnings" incl="faults"
  dn="uni/tn-tenant64/monepg-MonPoll/slsrc-VRF64_SyslogSource">
  <syslogRsDestGroup tDn="uni/fabric/slgroup-tenant64_SyslogDest"/>
</syslogSrc>
```

Enabling Syslog to Display in NX-OS CLI Format, Using the REST API

By default the Syslog format is RFC 3164 compliant. You can change the default display of Syslogs to NX-OS type format, similar to the following example:

```
apic1# moquery -c "syslogRemoteDest"
Total Objects shown: 1

# syslog.RemoteDest
host           : 172.23.49.77
adminState    : enabled
childAction   :
descr         :
dn            : uni/fabric/slgroup-syslog-mpod/rdst-172.23.49.77
epgDn         :
format        : nxos
forwardingFacility : local7
ip            :
lcOwn         : local
modTs         : 2016-05-17T16:51:57.231-07:00
monPolDn     : uni/fabric/monfab-default
name          : syslog-dest
operState     : unknown
port          : 514
rn            : rdst-172.23.49.77
severity      : information
status        :
uid           : 15374
vrfId         : 0
vrfName       :
```

To enable the Syslogs to display in NX-OS type format, perform the following steps, using the REST API.

Step 1 Enable the Syslogs to display in NX-OS type format, as in the following example:

```
POST https://192.168.20.123/api/node/mo/uni/fabric.xml
<syslogGroup name="DestGrp77" format="nxos">
<syslogRemoteDest name="slRmtDest77" host="172.31.138.20" severity="debugging"/>
</syslogGroup>
```

The **syslogGroup** is the Syslog monitoring destination group, the **sysLogRemoteDest** is the name you previously configured for your Syslog server, and the **host** is the IP address for the previously configured Syslog server.

Step 2 Set the Syslog format back to the default RFC 3164 format, as in the following example:

```
POST https://192.168.20.123/api/node/mo/uni/fabric.xml
<syslogGroup name="DestGrp77" format="aci">
<syslogRemoteDest name="slRmtDest77" host="172.31.138.20" severity="debugging"/>
</syslogGroup>
```

Data Plane Policing

Overview of Data Plane Policing

Use data plane policing (DPP) to manage bandwidth consumption on Cisco Application Centric Infrastructure (ACI) fabric access interfaces. DPP policies can apply to egress traffic, ingress traffic, or both. DPP monitors the data rates for a particular interface. When the data rate exceeds user-configured values, marking or dropping of packets occurs immediately. Policing does not buffer the traffic; therefore, the transmission delay is not affected. When traffic exceeds the data rate, the Cisco ACI fabric can either drop the packets or mark QoS fields in them.

Before the 3.2 release, the standard behavior for the policer was to be per-EPG member in the case of DPP policy being applied to the EPG, while the same policer was allocated on the leaf switch for the Layer 2 and Layer 3 case. This distinction was done because the DPP policer for Layer 2/Layer 3 case was assumed to be per-interface already, hence it was assumed different interfaces might get different ones. While the per-EPG DPP policy was introduced, it was clear that on a given leaf switch, several members could be present and therefore the policer it made sense to be per-member in order to avoid unwanted drops.

Starting with release 3.2, a clear semantic is given to the Data Plane Policer policy itself, as well as a new flag introducing the sharing-mode setting as presented in the CLI. Essentially, there is no longer an implicit behavior, which is different if the Data Plane Policer is applied to Layer 2/Layer 3 or to per-EPG case. Now the user has the control of the behavior. If the sharing-mode is set to **shared**, then all the entities on the leaf switch referring to the same Data Plane Policer, will share the same hardware policer. If the sharing-mode is set to **dedicated** then there would be a different HW policer allocated for each Layer 2 or Layer 3 or EPG member on the leaf switch. The policer is then dedicated to the entity that needs to be policed.

DPP policies can be single-rate, dual-rate, and color-aware. Single-rate policies monitor the committed information rate (CIR) of traffic. Dual-rate policers monitor both CIR and peak information rate (PIR) of traffic. In addition, the system monitors associated burst sizes. Three colors, or conditions, are determined by the policer for each packet depending on the data rate parameters supplied: conform (green), exceed (yellow), or violate (red).

Typically, DPP policies are applied to physical or virtual layer 2 connections for virtual or physical devices such as servers or hypervisors, and on layer 3 connections for routers. DPP policies applied to leaf switch access ports are configured in the fabric access (infra) portion of the Cisco ACI fabric, and must be configured by a fabric administrator. DPP policies applied to interfaces on border leaf switch access ports (l3extOut or l2extOut) are configured in the tenant (fvTenant) portion of the Cisco ACI fabric, and can be configured by a tenant administrator.

The data plane policer can also be applied on an EPG so that traffic that enters the Cisco ACI fabric from a group of endpoints are limited per member access interface of the EPG. This is useful to prevent monopolization of any single EPG where access links are shared by various EPGs.

Only one action can be configured for each condition. For example, a DPP policy can conform to the data rate of 256000 bits per second, with up to 200 millisecond bursts. The system applies the conform action to traffic that falls within this rate, and it would apply the violate action to traffic that exceeds this rate. Color-aware policies assume that traffic has been previously marked with a color. This information is then used in the actions taken by this type of policer.

For information about traffic storm control, see the *Cisco APIC Layer 2 Networking Configuration Guide*.

Configuring Data Plane Policing Using the REST API

To police the L2 traffic coming in to the Leaf:

```
<!-- api/node/mo/uni/.xml -->
<infraInfra>
<qosDppPol name="infradpp5" burst="2000" rate="2000" be="400" sharingMode="shared"/>
<!--
  List of nodes. Contains leaf selectors. Each leaf selector contains list of node blocks
-->
<infraNodeP name="leaf1">
<infraLeafS name="leaf1" type="range">
<infraNodeBlk name="leaf1" from_="101" to_="101"/>
</infraLeafS>
<infraRsAccPortP tDn="uni/infra/accportprof-portselector1"/>
</infraNodeP>
<!--
  PortP contains port selectors. Each port selector contains list of ports. It
  also has association to port group policies
-->
<infraAccPortP name="portselector1">
<infraHPortS name="pselc" type="range">
<infraPortBlk name="blk" fromCard="1" toCard="1" fromPort="48" toPort="49"></infraPortBlk>
<infraRsAccBaseGrp tDn="uni/infra/funcprof/accportgrp-portSet2"/>
</infraHPortS>
</infraAccPortP>
<!-- FuncP contains access bundle group policies -->
<infraFuncP>
<infraAccPortGrp name="portSet2">
<infraRsQosIngressDppIfPol tnQosDppPolName="infradpp5"/>
</infraAccPortGrp>
</infraFuncP>
</infraInfra>
```

To police the L2 traffic going out of the Leaf:

```
<!-- api/node/mo/uni/.xml -->
<infraInfra>
<qosDppPol name="infradpp2" burst="4000" rate="4000"/>
<!--
  List of nodes. Contains leaf selectors. Each leaf selector contains list of node blocks
-->
<infraNodeP name="leaf1">
<infraLeafS name="leaf1" type="range">
<infraNodeBlk name="leaf1" from_="101" to_="101"/>
</infraLeafS>
<infraRsAccPortP tDn="uni/infra/accportprof-portselector2"/>
</infraNodeP>
<!--
  PortP contains port selectors. Each port selector contains list of ports. It
  also has association to port group policies
-->
<infraAccPortP name="portselector2">
<infraHPortS name="pselc" type="range">
<infraPortBlk name="blk" fromCard="1" toCard="1" fromPort="37" toPort="38"></infraPortBlk>
<infraRsAccBaseGrp tDn="uni/infra/funcprof/accportgrp-portSet2"/>
</infraHPortS>
</infraAccPortP>
<!-- FuncP contains access bundle group policies -->
<infraFuncP>
<infraAccPortGrp name="portSet2">
<infraRsQosEgressDppIfPol tnQosDppPolName="infradpp2"/>
</infraAccPortGrp>
```

```
</infraFuncP>
</infraInfra>
```

To police the L3 traffic coming in to the Leaf:

```
<!-- api/node/mo/uni/.xml -->
<fvTenant name="dppTenant">
  <qosDppPol name="gmeo" burst="2000" rate="2000"/>
  <l3extOut name="Outside">
    <l3extInstP name="extroute"/>
    <l3extLNodeP name="borderLeaf">
      <l3extRsNodeL3OutAtt tDn="topology/pod-1/node-101" rtrId="10.0.0.1">
        <ipRouteP ip="0.0.0.0">
          <ipNextHopP nhAddr="192.168.62.2"/>
        </ipRouteP>
      </l3extRsNodeL3OutAtt>
      <l3extLIIfP name="portProfile">
        <l3extRsPathL3OutAtt addr="192.168.40.1/30" ifInstT="l3-port"
          tDn="topology/pod-1/paths-101/pathep-[eth1/40]"/>
        <l3extRsPathL3OutAtt addr="192.168.41.1/30" ifInstT="l3-port"
          tDn="topology/pod-1/paths-101/pathep-[eth1/41]"/>
        <l3extRsIngressQosDppPol tnQosDppPolName="gmeo"/>
      </l3extLIIfP>
    </l3extLNodeP>
  </l3extOut>
</fvTenant>
```

To police the L3 traffic going out of the Leaf:

```
<!-- api/node/mo/uni/.xml -->
<fvTenant name="dppTenant">
  <qosDppPol name="gmeo" burst="2000" rate="2000"/>
  <l3extOut name="Outside">
    <l3extInstP name="extroute"/>
    <l3extLNodeP name="borderLeaf">
      <l3extRsNodeL3OutAtt tDn="topology/pod-1/node-101" rtrId="10.0.0.1">
        <ipRouteP ip="0.0.0.0">
          <ipNextHopP nhAddr="192.168.62.2"/>
        </ipRouteP>
      </l3extRsNodeL3OutAtt>
      <l3extLIIfP name="portProfile">
        <l3extRsPathL3OutAtt addr="192.168.40.1/30" ifInstT="l3-port"
          tDn="topology/pod-1/paths-101/pathep-[eth1/40]"/>
        <l3extRsPathL3OutAtt addr="192.168.41.1/30" ifInstT="l3-port"
          tDn="topology/pod-1/paths-101/pathep-[eth1/41]"/>
        <l3extRsEgressQosDppPol tnQosDppPolName="gmeo"/>
      </l3extLIIfP>
    </l3extLNodeP>
  </l3extOut>
</fvTenant>
```

Traffic Storm Control

About Traffic Storm Control

A traffic storm occurs when packets flood the LAN, creating excessive traffic and degrading network performance. You can use traffic storm control policies to prevent disruptions on Layer 2 ports by broadcast, unknown multicast, or unknown unicast traffic storms on physical interfaces.

By default, storm control is not enabled in the ACI fabric. ACI bridge domain (BD) Layer 2 unknown unicast flooding is enabled by default within the BD but can be disabled by an administrator. In that case, a storm control policy only applies to broadcast and unknown multicast traffic. If Layer 2 unknown unicast flooding is enabled in a BD, then a storm control policy applies to Layer 2 unknown unicast flooding in addition to broadcast and unknown multicast traffic.

Traffic storm control (also called traffic suppression) allows you to monitor the levels of incoming broadcast, multicast, and unknown unicast traffic over a one second interval. During this interval, the traffic level, which is expressed either as percentage of the total available bandwidth of the port or as the maximum packets per second allowed on the given port, is compared with the traffic storm control level that you configured. When the ingress traffic reaches the traffic storm control level that is configured on the port, traffic storm control drops the traffic until the interval ends. An administrator can configure a monitoring policy to raise a fault when a storm control threshold is exceeded.

Configuring a Traffic Storm Control Policy Using the REST API

To configure a traffic storm control policy, create a `stormctrl:IfPol` object with the desired properties.

To create a policy named `MyStormPolicy`, send this HTTP POST message:

```
POST https://192.0.20.123/api/mo/uni/infra/stormctrlifp-MyStormPolicy.json
```

In the body of the POST message, include the following JSON payload structure to specify the policy by percentage of available bandwidth:

```
{ "stormctrlIfPol":
  { "attributes":
    { "dn": "uni/infra/stormctrlifp-MyStormPolicy",
      "name": "MyStormPolicy",
      "rate": "75",
      "burstRate": "85",
      "rn": "stormctrlifp-MyStormPolicy",
      "status": "created"
    },
    "children": []
  }
}
```

In the body of the POST message, include the following JSON payload structure to specify the policy by packets per second:

```
{ "stormctrlIfPol":
  { "attributes":
    { "dn": "uni/infra/stormctrlifp-MyStormPolicy",
      "name": "MyStormPolicy",
      "ratePps": "12000",
      "burstPps": "15000",
      "rn": "stormctrlifp-MyStormPolicy",
      "status": "created"
    },
    "children": []
  }
}
```

Rogue Endpoint Control

About the Rogue Endpoint Control Policy

A rogue endpoint attacks leaf switches through frequently, repeatedly injecting packets on different leaf switch ports and changing 802.1Q tags (thus, emulating endpoint moves) causing learned class and EPG port changes. Misconfigurations can also cause frequent IP and MAC address changes (moves).

Such rapid movement in the fabric causes significant network instability, high CPU usage, and in rare instances, endpoint mapper (EPM) and EPM client (EPMC) crashes due to significant and prolonged messaging and transaction service (MTS) buffer consumption. Also, such frequent moves may result in the EPM and EPMC logs rolling over very quickly, hampering debugging for unrelated endpoints.

The rogue endpoint control feature addresses this vulnerability by quickly:

- Identifying such rapidly moving MAC and IP endpoints.
- Stopping the movement by temporarily making endpoints static, thus quarantining the endpoint.
- Prior to 3.2(6) release: Keeping the endpoint static for the **Rogue EP Detection Interval** and dropping the traffic to and from the rogue endpoint. After this time expires, deleting the unauthorized MAC or IP address.
- In the 3.2(6) release and later: Keeping the endpoint static for the **Rogue EP Detection Interval** (this feature no longer drops the traffic). After this time expires, deleting the unauthorized MAC or IP address.
- Generating a host tracking packet to enable the system to re-learn the impacted MAC or IP address.
- Raising a fault to enable corrective action.

The rogue endpoint control policy is configured globally and, unlike other loop prevention methods, functions at the level of individual endpoints (IP and MAC addresses). It does not distinguish between local or remote moves; any type of interface change is considered a move in determining if an endpoint should be quarantined.

The rogue endpoint control feature is disabled by default.

Configure the Rogue Endpoint Control Policy Using the REST API

You can configure the rogue endpoint control policy for the fabric to detect and delete unauthorized endpoints using the REST API.

Step 1 To configure the rogue endpoint control policy, send a post with XML similar to the following example:

Example:

```
<polUni>
  <infraInfra>
    <epControlP name="default" adminSt="enabled" holdIntvl="1800" rogueEpDetectIntvl="60"
    rogueEpDetectMult="6"/>
  </infraInfra>
</polUni>
```

- `adminSt`: The administrative state of rogue endpoint control. Specify `enable` to enable rogue endpoint control.

- `holdIntvl`: Rogue endpoint hold interval. The hold interval is a period of time in seconds after the endpoint is declared rogue that the endpoint is kept static so that learning is prevented, and the traffic to and from the endpoint is dropped. After this interval, the endpoint is deleted. The valid values are from 1800 to 3600 seconds. The default is 1800.
- `rogueEpDetectIntvl`: Rogue endpoint detection interval. The detection interval is a period of time in seconds during which rogue endpoint control counts the number of moves for an endpoint. If the count during this interval exceeds the value specified by the detection multiplication factor, the endpoint is declared rogue. Valid values are from 0 to 65535 seconds. The default is 60.
- `rogueEpDetectMult`: Rogue endpoint detection multiplication factor. If an endpoint moves more times than the value specified by the detection multiplication factor during a period of time specified by the detection interval, the endpoint is declared rogue. Valid values are from 2 to 10. The default is 6.

Step 2

In the 3.2(6) release and later, you can revert this feature's behavior so that it once again drops the traffic to and from rogue endpoints by sending a post with XML similar to the following example:

Example:

```
<infraImplicitSetPol rogueModeAction="quarantine-fault-and-drop" infraDn="uni/infra"/>
```



CHAPTER 14

Provisioning Layer 2 Networks

- [Networking Domains, VLANs, and AEPs, on page 225](#)
- [Interfaces, on page 231](#)
- [FCoE, on page 243](#)
- [Fibre Channel NPV, on page 257](#)
- [802.1Q Tunnels, on page 264](#)
- [Breakout Ports, on page 271](#)
- [Port Profiles to Change Uplinks to Downlinks and Downlinks to Uplinks, on page 276](#)
- [IGMP Snooping, on page 281](#)
- [Proxy ARP, on page 285](#)
- [Flood on Encapsulation, on page 292](#)
- [MACsec, on page 297](#)

Networking Domains, VLANs, and AEPs

Networking Domains

A fabric administrator creates domain policies that configure ports, protocols, VLAN pools, and encapsulation. These policies can be used exclusively by a single tenant, or shared. Once a fabric administrator configures domains in the ACI fabric, tenant administrators can associate tenant endpoint groups (EPGs) to domains.

The following networking domain profiles can be configured:

- VMM domain profiles (`vmmDomP`) are required for virtual machine hypervisor integration.
- Physical domain profiles (`physDomP`) are typically used for bare metal server attachment and management access.
- Bridged outside network domain profiles (`l2extDomP`) are typically used to connect a bridged external network trunk switch to a leaf switch in the ACI fabric.
- Routed outside network domain profiles (`l3extDomP`) are used to connect a router to a leaf switch in the ACI fabric.
- Fibre Channel domain profiles (`fcDomP`) are used to connect Fibre Channel VLANs and VSANs.

A domain is configured to be associated with a VLAN pool. EPGs are then configured to use the VLANs associated with a domain.



Note EPG port and VLAN configurations must match those specified in the domain infrastructure configuration with which the EPG associates. If not, the APIC will raise a fault. When such a fault occurs, verify that the domain infrastructure configuration matches the EPG port and VLAN configurations.

Configuring a Physical Domain Using the REST API

A physical domain acts as the link between the VLAN pool and the Access Entity Profile (AEP). The domain also ties the fabric configuration to the tenant configuration, as the tenant administrator is the one who associates domains to EPGs, while the domains are created under the fabric tab. When configuring in this order, only the profile name and the VLAN pool are configured.

Configure a physical domain by sending a post with XML such as the following example:

Example:

```
<physDomP dn="uni/phys-bsprint-PHY" lcOwn="local" modTs="2015-02-23T16:13:21.906-08:00"
  monPolDn="uni/fabric/monfab-default" name="bsprint-PHY" ownerKey="" ownerTag="" status="" uid="8131">
  <infraRsVlanNs childAction="" forceResolve="no" lcOwn="local"
modTs="2015-02-23T16:13:22.065-08:00"
  monPolDn="uni/fabric/monfab-default" rType="mo" rn="rsvlanNs" state="formed" stateQual="none"
  status="" tCl="fvnsVlanInstP" tDn="uni/infra/vlanns-[bsprint-vlan-pool]-static" tType="mo"
uid="8131"/>
  <infraRsVlanNsDef forceResolve="no" lcOwn="local" modTs="2015-02-23T16:13:22.065-08:00" rType="mo"
  rn="rsvlanNsDef" state="formed" stateQual="none" status="" tCl="fvnsAInstP"
  tDn="uni/infra/vlanns-[bsprint-vlan-pool]-static" tType="mo"/>
  <infraRtDomP lcOwn="local" modTs="2015-02-23T16:13:52.945-08:00"
rn="rtDomP-[uni/infra/attentp-bsprint-AEP]"
  status="" tCl="infraAttEntityP" tDn="uni/infra/attentp-bsprint-AEP"/>
</physDomP>
```

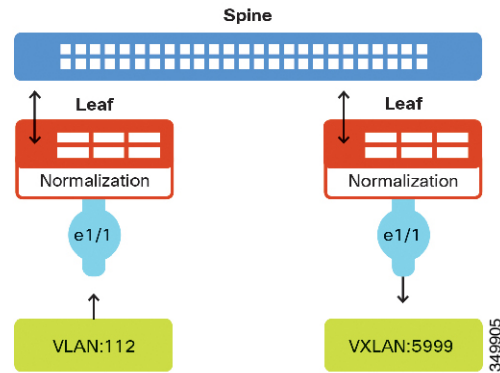
Creating VLAN Pools

In this example, configuring newly-connected bare metal servers first requires creation of a physical domain and then association of the domain to a VLAN pool. As mentioned in the previous section, VLAN pools define a range of VLAN IDs that will be used by the EPGs.

The servers are connected to two different leaf nodes in the fabric. Each server will be tagging using 802.1Q or VXLAN encapsulation. The range of VLANs used in the configuration example is 100-199. As depicted in the following figure, the ACI fabric can also act as a gateway between disparate encapsulation types such as untagged traffic, 802.1Q VLAN tags, VXLAN VNIDs, and NVGRE tags. The leaf switches normalize the traffic by stripping off tags and reapplying the required tags on fabric egress. In ACI, it is important to understand that the definition of VLANs as they pertain to the leaf switch ports is utilized only for identification purposes. When a packet arrives ingress to a leaf switch in the fabric, ACI has to know beforehand how to

classify packets into the different EPGs, using identifiers like VLANs, VXLAN, NVGRE, physical port IDs, virtual port IDs.

Figure 21: Encapsulation normalization



Creating a VLAN Pool Using the REST API

The following example REST request creates a VLAN pool:

```
<fvnsVlanInstP allocMode="static" childAction="" configIssues="" descr=""
  dn="uni/infra/vlanns-[bsprint-vlan-pool]-static" lcOwn="local"
  modTs="2015-02-23T15:58:33.538-08:00"
  monPolDn="uni/fabric/monfab-default" name="bsprint-vlan-pool"
  ownerKey="" ownerTag="" status="" uid="8131">
  <fvnsRtVlanNs childAction="" lcOwn="local" modTs="2015-02-25T11:35:33.365-08:00"
    rn="rtinfraVlanNs-[uni/l2dom-JC-L2-Domain]" status="" tCl="l2extDomP"
  tDn="uni/l2dom-JC-L2-Domain"/>
  <fvnsRtVlanNs childAction="" lcOwn="local" modTs="2015-02-23T16:13:22.007-08:00"
    rn="rtinfraVlanNs-[uni/phys-bsprint-PHY]" status="" tCl="physDomP"
  tDn="uni/physbsprint-PHY"/>
  <fvnsEncapBlk childAction="" descr="" from="vlan-100" lcOwn="local"
  modTs="2015-02-23T15:58:33.538-08:00"
    name="" rn="from-[vlan-100]-to-[vlan-199]" status="" to="vlan-199" uid="8131"/>
</fvnsVlanInstP>
```

Configuring Q-in-Q Encapsulation Mapping for EPGs

Q-in-Q Encapsulation Mapping for EPGs

Using Cisco Application Policy Infrastructure Controller (APIC), you can map double-tagged VLAN traffic ingressing on a regular interface, PC, or vPC to an EPG. When this feature is enabled, when double-tagged traffic enters the network for an EPG, both tags are processed individually in the fabric and restored to double-tags when egressing the Cisco Application Centric Infrastructure (ACI) switch. Ingressing single-tagged and untagged traffic is dropped.

The following guidelines and limitations apply:

- This feature is only supported on Cisco Nexus 9300-FX platform switches.
- Both the outer and inner tag must be of EtherType 0x8100.

- MAC learning and routing are based on the EPG port, sclass, and VRF instance, not on the access encapsulations.
- QoS priority settings are supported, derived from the outer tag on ingress, and rewritten to both tags on egress.
- EPGs can simultaneously be associated with other interfaces on a leaf switch, that are configured for single-tagged VLANs.
- Service graphs are supported for provider and consumer EPGs that are mapped to Q-in-Q encapsulated interfaces. You can insert service graphs, as long as the ingress and egress traffic on the service nodes is in single-tagged encapsulated frames.
- When vPC ports are enabled for Q-in-Q encapsulation mode, VLAN consistency checks are not performed.

The following features and options are not supported with this feature:

- Per-port VLAN feature
- FEX connections
- Mixed mode

For example, an interface in Q-in-Q encapsulation mode can have a static path binding to an EPG with double-tagged encapsulation only, not with regular VLAN encapsulation.
- STP and the "Flood in Encapsulation" option
- Untagged and 802.1p mode
- Multi-pod and Multi-Site
- Legacy bridge domain
- L2Out and L3Out connections
- VMM integration
- Changing a port mode from routed to Q-in-Q encapsulation mode
- Per-VLAN mis-cabling protocol on ports in Q-in-Q encapsulation mode

Mapping EPGs to Q-in-Q Encapsulation Enabled Interfaces Using the REST API

Before you begin

Create the tenant, application profile, and application EPG that will be mapped with an interface configured for Q-in-Q mode.

SUMMARY STEPS

1. Enable an interface for Q-in-Q encapsulation and associate the interface with an EPG, with XML such as the following example:

DETAILED STEPS

Enable an interface for Q-in-Q encapsulation and associate the interface with an EPG, with XML such as the following example:

Example:

```
<polUni>
  <fvTenant dn="uni/tn-tenant64" name="tenant64">
    <fvCtx name="VRF64"/>
    <fvBD name="BD64_1">
      <fvRsCtx tnFvCtxName="VRF64"/>
      <fvSubnet ip="20.0.1.2/24"/>
    </fvBD>
    <fvAp name="AP64">
      <fvAEPg name="WEB7">
        <fvRsBd tnFvBDName="BD64_1"/>
        <fvRsQinqPathAtt tDn="topology/pod-1/paths-101/pathep-[eth1/25]" encap="qinq-202-203"/>
      </fvAEPg>
    </fvAp>
  </fvTenant>
</polUni>
```

Attachable Entity Profile

The ACI fabric provides multiple attachment points that connect through leaf ports to various external entities such as bare metal servers, virtual machine hypervisors, Layer 2 switches (for example, the Cisco UCS fabric interconnect), or Layer 3 routers (for example Cisco Nexus 7000 Series switches). These attachment points can be physical ports, FEX ports, port channels, or a virtual port channel (vPC) on leaf switches.



Note When creating a VPC domain between two leaf switches, both switches must be in the same switch generation, one of the following:

- Generation 1 - Cisco Nexus N9K switches without “EX” or “FX” on the end of the switch name; for example, N9K-9312TX
- Generation 2 – Cisco Nexus N9K switches with “EX” or “FX” on the end of the switch model name; for example, N9K-93108TC-EX

Switches such as these two are not compatible VPC peers. Instead, use switches of the same generation.

An Attachable Entity Profile (AEP) represents a group of external entities with similar infrastructure policy requirements. The infrastructure policies consist of physical interface policies that configure various protocol options, such as Cisco Discovery Protocol (CDP), Link Layer Discovery Protocol (LLDP), or Link Aggregation Control Protocol (LACP).

An AEP is required to deploy VLAN pools on leaf switches. Encapsulation blocks (and associated VLANs) are reusable across leaf switches. An AEP implicitly provides the scope of the VLAN pool to the physical infrastructure.

The following AEP requirements and dependencies must be accounted for in various configuration scenarios, including network connectivity, VMM domains, and multipod configuration:

- The AEP defines the range of allowed VLANs but it does not provision them. No traffic flows unless an EPG is deployed on the port. Without defining a VLAN pool in an AEP, a VLAN is not enabled on the leaf port even if an EPG is provisioned.
- A particular VLAN is provisioned or enabled on the leaf port that is based on EPG events either statically binding on a leaf port or based on VM events from external controllers such as VMware vCenter or Microsoft Azure Service Center Virtual Machine Manager (SCVMM).
- Attached entity profiles can be associated directly with application EPGs, which deploy the associated application EPGs to all those ports associated with the attached entity profile. The AEP has a configurable generic function (infraGeneric), which contains a relation to an EPG (infraRsFuncToEpg) that is deployed on all interfaces that are part of the selectors that are associated with the attachable entity profile.

A virtual machine manager (VMM) domain automatically derives physical interface policies from the interface policy groups of an AEP.

An override policy at the AEP can be used to specify a different physical interface policy for a VMM domain. This policy is useful in scenarios where a VM controller is connected to the leaf switch through an intermediate Layer 2 node, and a different policy is desired at the leaf switch and VM controller physical ports. For example, you can configure LACP between a leaf switch and a Layer 2 node. At the same time, you can disable LACP between the VM controller and the Layer 2 switch by disabling LACP under the AEP override policy.

Creating an Attachable Access Entity Profile Using the REST API

The following example REST request creates an attachable access entity profile (AEP):

```
<infraAttEntityP childAction="" configIssues="" descr="" dn="uni/infra/attentpbsprint-AEP"
  lcOwn="local" modTs="2015-02-23T16:13:52.874-08:00" monPolDn="uni/fabric/monfab-default"
  name="bsprint-AEP" ownerKey="" ownerTag="" status="" uid="8131">
  <infraContDomP childAction="" lcOwn="local" modTs="2015-02-23T16:13:52.874-08:00"
    rn="dompcont" status="">
    <infraAssocDomP childAction="" dompDn="uni/phys-bsprint-PHY" lcOwn="local"
      modTs="2015-02-23T16:13:52.961-08:00" rn="assocdomp-[uni/phys-bsprint-PHY]"
      status=""/>
    <infraAssocDomP childAction="" dompDn="uni/l2dom-JC-L2-Domain" lcOwn="local"
      modTs="2015-02-25T11:35:33.570-08:00" rn="assocdomp-[uni/l2dom-JC-L2-Domain]"
      status=""/>
  </infraContDomP>
  <infraContNS childAction="" lcOwn="local" modTs="2015-02-23T16:13:52.874-08:00"
    monPolDn="uni/fabric/monfab-default" rn="nscont" status="">
    <infraRsToEncapInstDef childAction="" deplSt="" forceResolve="no" lcOwn="local"
      modTs="2015-02-23T16:13:52.961-08:00" monPolDn="uni/fabric/monfabdefault"
      rType="mo" rn="rstoEncapInstDef-[allocencap-[uni/infra]/encapnsdef-
      [uni/infra/vlanns-[bsprint-vlan-pool]-static]]" state="formed" stateQual="none"
      status="" tCl="stpEncapInstDef" tDn="allocencap-[uni/infra]/encapnsdef-
      [uni/infra/vlanns-[bsprint-vlan-pool]-static]" tType="mo">
      <fabricCreatedBy childAction="" creatorDn="uni/l2dom-JC-L2-Domain"
        deplSt="" domainDn="uni/l2dom-JC-L2-Domain" lcOwn="local" modTs="2015-02-
        25T11:35:33.570-08:00" monPolDn="uni/fabric/monfab-default" profileDn=""
        rn="source-[uni/l2dom-JC-L2-Domain]" status=""/>
      <fabricCreatedBy childAction="" creatorDn="uni/phys-bsprint-PHY" deplSt=""
        domainDn="uni/phys-bsprint-PHY" lcOwn="local"
        modTs="2015-02-23T16:13:52.961-08:00"
        monPolDn="uni/fabric/monfab-default" profileDn=""
        rn="source-[uni/phys-bsprint-PHY]"
        status=""/>
    </infraRsToEncapInstDef>
  </infraContNS>
  <infraRtAttEntP childAction="" lcOwn="local" modTs="2015-02-24T11:59:37.980-08:00"
```



```

    rn="rtattEntP-[uni/infra/funcprof/accportgrp-bsprint-AccessPort]" status=""
    tCl="infraAccPortGrp" tDn="uni/infra/funcprof/accportgrp-bsprint-AccessPort"/>
<infraRsDomP childAction="" forceResolve="no" lcOwn="local" modTs="2015-02-
25T11:35:33.570-08:00" monPolDn="uni/fabric/monfab-default" rType="mo"
    rn="rsdomP-[uni/l2dom-JC-L2-Domain]" state="formed" stateQual="none" status=""
    tCl="l2extDomP" tDn="uni/l2dom-JC-L2-Domain" tType="mo" uid="8754"/>
<infraRsDomP childAction="" forceResolve="no" lcOwn="local"
    modTs="2015-02-23T16:13:52.961-08:00" monPolDn="uni/fabric/monfab-default" rType="mo"

    rn="rsdomP-[uni/phys-bsprint-PHY]" state="formed" stateQual="none" status=""
    tCl="physDomP"
    tDn="uni/phys-bsprint-PHY" tType="mo" uid="8131"/>
</infraAttEntityP>

```

Interfaces

Ports, PCs, and VPCs

Configuring a Single Port Channel Applied to Multiple Switches

This example creates a port channel on leaf switch 17, another port channel on leaf switch 18, and a third one on leaf switch 20. On each leaf switch, the same interfaces will be part of the port channel (interfaces 1/10 to 1/15 and 1/20 to 1/25). All these port channels will have the same configuration.

Before you begin

- The ACI fabric is installed, APIC controllers are online, and the APIC cluster is formed and healthy.
- An APIC fabric administrator account is available that will enable creating the necessary fabric infrastructure configurations.
- The target leaf switch and protocol(s) are configured and available.

To create the port channel, send a post with XML such as the following:

Example:

```

<infraInfra dn="uni/infra">
  <infraNodeP name="test">
    <infraLeafS name="leafs" type="range">
      <infraNodeBlk name="nblk" from_="17" to_="18"/>
      <infraNodeBlk name="nblk" from_="20" to_="20"/>
    </infraLeafS>
    <infraRsAccPortP tDn="uni/infra/accportprof-test"/>
  </infraNodeP>

  <infraAccPortP name="test">
    <infraHPortS name="pselc" type="range">
      <infraPortBlk name="blk1"
        fromCard="1" toCard="1"
        fromPort="10" toPort="15"/>
      <infraPortBlk name="blk2"
        fromCard="1" toCard="1"
        fromPort="20" toPort="25"/>
    </infraHPortS>
  </infraAccPortP>
</infraInfra>

```

```

        <infraRsAccBaseGrp
            tDn="uni/infra/funcprof/accbundle-bndlgrp"/>
    </infraHPorts>
</infraAccPortP>

<infraFuncP>
    <infraAccBndlGrp name="bndlgrp" lagT="link">
        <infraRsHIfPol tnFabricHIfPolName="default"/>
        <infraRsCdpIfPol tnCdpIfPolName="default"/>
        <infraRsLacpPol tnLacpLagPolName="default"/>
    </infraAccBndlGrp>
</infraFuncP>

</infraInfra>

```

Configuring a Single Virtual Port Channel Across Two Switches Using the REST API

The two steps for creating a virtual port channel across two switches are as follows:

- Create a `fabricExplicitGep`: this policy specifies the leaf switch that pairs to form the virtual port channel.
- Use the `infra` selector to specify the interface configuration.

The APIC performs several validations of the `fabricExplicitGep` and faults are raised when any of these validations fail. A leaf can be paired with only one other leaf. The APIC rejects any configuration that breaks this rule. When creating a `fabricExplicitGep`, an administrator must provide the IDs of both of the leaf switches to be paired. The APIC rejects any configuration which breaks this rule. Both switches must be up when `fabricExplicitGep` is created. If one switch is not up, the APIC accepts the configuration but raises a fault. Both switches must be leaf switches. If one or both switch IDs corresponds to a spine, the APIC accepts the configuration but raises a fault.

Before you begin

- The ACI fabric is installed, APIC controllers are online, and the APIC cluster is formed and healthy.
- An APIC fabric administrator account is available that will enable creating the necessary fabric infrastructure configurations.
- The target leaf switch and protocol(s) are configured and available.

To create the `fabricExplicitGep` policy and use the `infra` selector to specify the interface, send a post with XML such as the following example:

Example:

```

<fabricProtPol pairT="explicit">
<fabricExplicitGep name="tG" id="2">
    <fabricNodePEp id="18"/>
    <fabricNodePEp id="25"/>
    </fabricExplicitGep>
</fabricProtPol>

```

Configuring Two Port Channels Applied to Multiple Switches Using the REST API

This example creates two port channels (PCs) on leaf switch 17, another port channel on leaf switch 18, and a third one on leaf switch 20. On each leaf switch, the same interfaces will be part of the PC (interface 1/10 to 1/15 for port channel 1 and 1/20 to 1/25 for port channel 2). The policy uses two switch blocks because each a switch block can contain only one group of consecutive switch IDs. All these PCs will have the same configuration.



Note Even though the PC configurations are the same, this example uses two different interface policy groups. Each Interface Policy Group represents a PC on a switch. All interfaces associated with a given interface policy group are part of the same PCs.

Before you begin

- The ACI fabric is installed, APIC controllers are online, and the APIC cluster is formed and healthy.
- An APIC fabric administrator account is available that will enable creating the necessary fabric infrastructure configurations.
- The target leaf switch and protocol(s) are configured and available.

To create the two PCs, send a post with XML such as the following:

Example:

```
<infraInfra dn="uni/infra">
  <infraNodeP name="test">
    <infraLeafS name="leafs" type="range">
      <infraNodeBlk name="nblk"
        from_"17" to_"18"/>
      <infraNodeBlk name="nblk"
        from_"20" to_"20"/>
    </infraLeafS>
    <infraRsAccPortP tDn="uni/infra/accportprof-test1"/>
    <infraRsAccPortP tDn="uni/infra/accportprof-test2"/>
  </infraNodeP>

  <infraAccPortP name="test1">
    <infraHPortS name="pselc" type="range">
      <infraPortBlk name="blk1"
        fromCard="1" toCard="1"
        fromPort="10" toPort="15"/>
    <infraRsAccBaseGrp
      tDn="uni/infra/funcprof/accbundle-bndlgrp1"/>
    </infraHPortS>
  </infraAccPortP>

  <infraAccPortP name="test2">
    <infraHPortS name="pselc" type="range">
      <infraPortBlk name="blk1"
        fromCard="1" toCard="1"
        fromPort="20" toPort="25"/>
    <infraRsAccBaseGrp
      tDn="uni/infra/funcprof/accbundle-bndlgrp2" />
    </infraHPortS>
</infraInfra>
```

```

</infraAccPortP>

<infraFuncP>
  <infraAccBndlGrp name="bndlgrp1" lagT="link">
    <infraRsHIfPol tnFabricHIfPolName="default"/>
    <infraRsCdpIfPol tnCdpIfPolName="default"/>
    <infraRsLacpPol tnLacpLagPolName="default"/>
  </infraAccBndlGrp>

  <infraAccBndlGrp name="bndlgrp2" lagT="link">
    <infraRsHIfPol tnFabricHIfPolName="default"/>
    <infraRsCdpIfPol tnCdpIfPolName="default"/>
    <infraRsLacpPol tnLacpLagPolName="default"/>
  </infraAccBndlGrp>
</infraFuncP>

</infraInfra>

```

Configuring a Virtual Port Channel on Selected Port Blocks of Two Switches Using the REST API

This policy creates a single virtual port channel (vPC) on leaf switches 18 and 25, using interfaces 1/10 to 1/15 on leaf 18, and interfaces 1/20 to 1/25 on leaf 25.

Before you begin

- The ACI fabric is installed, APIC controllers are online, and the APIC cluster is formed and healthy.
- An APIC fabric administrator account is available that will enable creating the necessary fabric infrastructure configurations.
- The target leaf switch and protocol(s) are configured and available.



Note When creating a vPC domain between two leaf switches, both switches must be in the same switch generation, one of the following:

- Generation 1 - Cisco Nexus N9K switches without “EX” on the end of the switch name; for example, N9K-9312TX
- Generation 2 – Cisco Nexus N9K switches with “EX” on the end of the switch model name; for example, N9K-93108TC-EX

Switches such as these two are not compatible vPC peers. Instead, use switches of the same generation.

To create the vPC send a post with XML such as the following example:

Example:

```

<infraInfra dn="uni/infra">

  <infraNodeP name="test1">
    <infraLeafS name="leafs" type="range">
      <infraNodeBlk name="nblk"
        from_"18" to_"18"/>
    </infraLeafS>
  </infraNodeP>
</infraInfra>

```

```

        </infraLeafS>
        <infraRsAccPortP tDn="uni/infra/accportprof-test1"/>
</infraNodeP>

<infraNodeP name="test2">
  <infraLeafS name="leafs" type="range">
    <infraNodeBlk name="nblk"
      from_="25" to_="25"/>
  </infraLeafS>
  <infraRsAccPortP tDn="uni/infra/accportprof-test2"/>
</infraNodeP>

<infraAccPortP name="test1">
  <infraHPortS name="pselc" type="range">
    <infraPortBlk name="blk1"
      fromCard="1" toCard="1"
      fromPort="10" toPort="15"/>
    <infraRsAccBaseGrp
      tDn="uni/infra/funcprof/accbundle-bndlgrp" />
  </infraHPortS>
</infraAccPortP>

<infraAccPortP name="test2">
  <infraHPortS name="pselc" type="range">
    <infraPortBlk name="blk1"
      fromCard="1" toCard="1"
      fromPort="20" toPort="25"/>
    <infraRsAccBaseGrp
      tDn="uni/infra/funcprof/accbundle-bndlgrp" />
  </infraHPortS>
</infraAccPortP>

<infraFuncP>
  <infraAccBndlGrp name="bndlgrp" lagT="node">
    <infraRsHIfPol tnFabricHIfPolName="default"/>
    <infraRsCdpIfPol tnCdpIfPolName="default"/>
    <infraRsLacpPol tnLacpLagPolName="default"/>
  </infraAccBndlGrp>
</infraFuncP>

</infraInfra>

```

Configuring a Virtual Port Channel and Applying it to a Static Port Using the REST API

Before you begin

- Install the APIC, verify that the APIC controllers are online, and that the APIC cluster is formed and healthy.
- Verify that an APIC fabric administrator account is available that will enable you to create the necessary fabric infrastructure.
- Verify that the target leaf switches are registered in the ACI fabric and available.

Step 1 To build vPCs, send a post with XML such as the following example:

Example:

```

https://apic-ip-address/api/policymgr/mo/.xml
<polUni>
<infraInfra>
<infraNodeP name="switchProfileforVPC_201">
<infraLeafS name="switchProfileforVPC_201" type="range">
<infraNodeBlk name="nodeBlk" from_="201" to_="201"/>
</infraLeafS>
<infraRsAccPortP tDn="uni/infra/accportprof-intProfileforVPC_201"/>
</infraNodeP>
<infraNodeP name="switchProfileforVPC_202">
<infraLeafS name="switchProfileforVPC_202" type="range">
<infraNodeBlk name="nodeBlk" from_="202" to_="202"/>
</infraLeafS>
<infraRsAccPortP tDn="uni/infra/accportprof-intProfileforVPC_202"/>
</infraNodeP>
<infraAccPortP name="intProfileforVPC_201">
<infraHPortS name="vpc201-202" type="range">
<infraPortBlk name="vpcPort1-15" fromCard="1" toCard="1" fromPort="15"
toPort="15"/>
<infraRsAccBaseGrp tDn="uni/infra/funcprof/accbundle-intPolicyGroupforVPC"/>
</infraHPortS>
</infraAccPortP>
<infraAccPortP name="intProfileforVPC_202">
<infraHPortS name="vpc201-202" type="range">
<infraPortBlk name="vpcPort1-1" fromCard="1" toCard="1" fromPort="1"
toPort="1"/>
<infraRsAccBaseGrp tDn="uni/infra/funcprof/accbundle-intPolicyGroupforVPC"/>
</infraHPortS>
</infraAccPortP>
<infraFuncP>
<infraAccBndlGrp name="intPolicyGroupforVPC" lagT="node">
<infraRsAttEntP tDn="uni/infra/attentp-AttEntityProfileforCisco"/>
<infraRsCdpIfPol tnCdpIfPolName="CDP_ON" />
<infraRsLacpPol tnLacpLagPolName="LACP_ACTIVE" />
<infraRsHIfPol tnFabricHIfPolName="10GigAuto" />
</infraAccBndlGrp>
</infraFuncP>
</infraInfra>
</polUni>

```

Step 2 To attach the VPC to static port bindings, send a post with XML such as the following:

Example:

```

https://apic-ip-address/api/node/mo/uni.xml
<polUni>
<fvTenant dn="uni/tn-Cisco" name="Cisco" ownerKey="" ownerTag="">
<fvAp name="CCO" ownerKey="" ownerTag="" prio="unspecified">
<fvAEPg matchT="AtleastOne" name="Web" prio="unspecified">
<fvRsPathAtt encap="vlan-1201" instrImedcy="immediate" mode="native"
tDn="topology/pod-1/protopaths-201-202/pathep-[vpc201-202]" />
</fvAEPg>
<fvAEPg matchT="AtleastOne" name="App" prio="unspecified">
<fvRsPathAtt encap="vlan-1202" instrImedcy="immediate" mode="native"
tDn="topology/pod-1/protopaths-201-202/pathep-[vpc201-202]" />
</fvAEPg>
</fvAp>
</fvTenant>
</polUni>

```

Reflective Relay (802.1Qbg)

Reflective relay is a switching option beginning with Cisco APIC Release 2.3(1). Reflective relay—the tagless approach of IEEE standard 802.1Qbg—forwards all traffic to an external switch, which then applies policy and sends the traffic back to the destination or target VM on the server as needed. There is no local switching. For broadcast or multicast traffic, reflective relay provides packet replication to each VM locally on the server.

One benefit of reflective relay is that it leverages the external switch for switching features and management capabilities, freeing server resources to support the VMs. Reflective relay also allows policies that you configure on the Cisco APIC to apply to traffic between the VMs on the same server.

In the Cisco ACI, you can enable reflective relay, which allows traffic to turn back out of the same port it came in on. You can enable reflective relay on individual ports, port channels, or virtual port channels as a Layer 2 interface policy using the APIC GUI, NX-OS CLI, or REST API. It is disabled by default.

The term *Virtual Ethernet Port Aggregator (VEPA)* is also used to describe 802.1Qbg functionality.

Reflective Relay Support

Reflective relay supports the following:

- IEEE standard 802.1Qbg tagless approach, known as reflective relay.

Cisco APIC Release 2.3(1) release does not support the IEEE standard 802.1Qbg S-tagged approach with multichannel technology.

- Physical domains.

Virtual domains are not supported.

- Physical ports, port channels (PCs), and virtual port channels (vPCs).

Cisco Fabric Extender (FEX) and blade servers are not supported. If reflective relay is enabled on an unsupported interface, a fault is raised, and the last valid configuration is retained. Disabling reflective relay on the port clears the fault.

- Cisco Nexus 9000 series switches with *EX* or *FX* at the end of their model name.

Enabling Reflective Relay Using the REST API

Reflective relay is disabled by default; however, you can enable it on a port, port channel, or virtual port channel as a Layer 2 interface policy on the switch.

Before you begin

This procedure assumes that you have set up the Cisco Application Centric Infrastructure (ACI) fabric and installed the physical switches.

Step 1 Configure a Layer 2 Interface policy with reflective relay enabled.

Example:

```
<l2IfPol name="VepaL2IfPol" vepa="enabled" />
```

Step 2 Apply the Layer 2 interface policy to a leaf access port policy group.

Example:

```
<infraAccPortGrp name="VepaPortG">
  <infraRsL2IfPol tnL2IfPolName="VepaL2IfPol"/>
</infraAccPortGrp>
```

Step 3 Configure an interface profile with an interface selector.

Example:

```
<infraAccPortP name="vepa">
  <infraHPortS name="pselc" type="range">
    <infraPortBlk name="blk"
      fromCard="1" toCard="1" fromPort="20" toPort="22">
    </infraPortBlk>
    <infraRsAccBaseGrp tDn="uni/infra/funcprof/accportgrp-VepaPortG" />
  </infraHPortS>
</infraAccPortP>
```

Step 4 Configure a node profile with node selector.

Example:

```
<infraNodeP name="VepaNodeProfile">
  <infraLeafS name="VepaLeafSelector" type="range">
    <infraNodeBlk name="VepaNodeBlk" from_="101" to_="102"/>
  </infraLeafS>
  <infraRsAccPortP tDn="uni/infra/accportprof-vepa"/>
</infraNodeP>
```

Interface Speed

Interface Configuration Guidelines

When configuring interfaces in a Cisco Application Centric Infrastructure (ACI) fabric, follow these guidelines.

Half duplex at 100Mbps speed is not supported

In a Cisco ACI leaf switch that supports 100Mbps speed, the 100Mbps speed is supported only if the link is in full duplex mode and if autonegotiation is configured the same on both the local and remote peer. The Cisco ACI leaf switch and the remote link should both be configured in full duplex mode with autonegotiation disabled on both devices or enabled on both devices.

Connecting an SFP module requires a link speed policy

When you connect an SFP module to a new card, you must create a link speed policy for the module to communicate with the card. Follow these steps to create a link speed policy.

1. Create an interface policy to specify the link speed, as in this example:

```
<fabricHIfPol name="mySpeedPol" speed="1G"/>
```

2. Reference the link speed policy within an interface policy group, as in this example:

```
<infraAccPortGrp name="myGroup">
  <infraRsHIfPol tnFabricHIfPolName="SpeedPol"/>
</infraAccPortGrp>
```


MAC Pinning

MAC pinning is used for pinning VM traffic in a round-robin fashion to each uplink based on the MAC address of the VM. In a normal virtual port channel (vPC), a hash algorithm uses the source and destination MAC address to determine which uplink will carry a packet. In a vPC with MAC pinning, VM1 might be pinned to the first uplink, VM2 pinned to the second uplink, and so on.

MAC pinning is the recommended option for channeling when connecting to upstream switches that do not support Multichassis EtherChannel (MEC).

Consider these guidelines and restrictions when configuring MAC pinning:

- When a Cisco Application Virtual Switch or Cisco ACI Virtual Edge is deployed behind a vPC with MAC pinning and a host is connected to two leaf switches using that same vPC, reloading one of the two leaf switches can result in a few minutes of traffic disruption.
- In the API, MAC pinning is selected in the LACP policy by setting `lacp:LagPol:mode` to `mac-pin`. When the policy is applied to a vPC, the vPC status as shown in `pc:AggrIf:pcMode` and in `pc:AggrIf:operChannelMode` is displayed as `active`, not as `mac-pin`.

Changing Interface Speed

This task creates a policy that configures the speed for a set of interfaces.

To set the speed for a set of interfaces, send a post with XML such as the following example:

Example:

```
<infraInfra dn="uni/infra">
  <infraNodeP name="test1">
    <infraLeafS name="leafs" type="range">
      <infraNodeBlk name="nblk" from_"18" to_"18"/>
    </infraLeafS>
    <infraRsAccPortP tDn="uni/infra/accportprof-test1"/>
  </infraNodeP>
  <infraNodeP name="test2">
    <infraLeafS name="leafs" type="range">
      <infraNodeBlk name="nblk" from_"25" to_"25"/>
    </infraLeafS>
    <infraRsAccPortP tDn="uni/infra/accportprof-test2"/>
  </infraNodeP>
  <infraAccPortP name="test1">
    <infraHPortS name="pselc" type="range">
      <infraPortBlk name="blk1"
        fromCard="1" toCard="1"
        fromPort="10" toPort="15"/>
    <infraRsAccBaseGrp
      tDn="uni/infra/funcprof/accbundle-bndlgrp" />
    </infraHPortS>
  </infraAccPortP>
  <infraAccPortP name="test2">
    <infraHPortS name="pselc" type="range">
      <infraPortBlk name="blk1"
        fromCard="1" toCard="1"
        fromPort="20" toPort="25"/>
    <infraRsAccBaseGrp
```

```

        tDn="uni/infra/funcprof/accbundle-bndlgrp" />
    </infraHPortS>
</infraAccPortP>

<infraFuncP>
    <infraAccBndlGrp name="bndlgrp" lagT="node">
        <infraRsHifPol tnFabricHifPolName="default"/>
        <infraRsCdpIfPol tnCdpIfPolName="default"/>
        <infraRsLacpPol tnLacpLagPolName="default"/>
    </infraAccBndlGrp>
</infraFuncP>

</infraInfra>

```

FEXs

ACI FEX Guidelines

Observe the following guidelines when deploying a FEX:

- Assuming that no leaf switch front panel ports are configured to deploy and EPG and VLANs, a maximum of 10,000 port EPGs are supported for being deployed using a FEX.
- For each FEX port or vPC that includes FEX ports as members, a maximum of 20 EPGs per VLAN are supported.
- A vPC with FEX interfaces ignores the minimum and maximum number of links configured in its port-channel policy. The vPC remains up even if the number of links is less than the minimum or greater than the maximum.

Configuring an FEX VPC Policy Using the REST API

This task creates a FEX virtual port channel (VPC) policy.

Before you begin

- The ACI fabric is installed, APIC controllers are online, and the APIC cluster is formed and healthy.
- An APIC fabric administrator account is available that will enable creating the necessary fabric infrastructure configurations.
- The target leaf switch, interfaces, and protocol(s) are configured and available.
- The FEXes are configured, powered on, and connected to the target leaf interfaces



Note When creating a VPC domain between two leaf switches, both switches must be in the same switch generation, one of the following:

- Generation 1 - Cisco Nexus N9K switches without “EX” on the end of the switch name; for example, N9K-9312TX
- Generation 2 – Cisco Nexus N9K switches with “EX” on the end of the switch model name; for example, N9K-93108TC-EX

Switches such as these two are not compatible VPC peers. Instead, use switches of the same generation.

To create the policy linking the FEX through a VPC to two switches, send a post with XML such as the following example:

Example:

```
<polUni>
<infraInfra dn="uni/infra">

<infraNodeP name="fexNodeP105">
  <infraLeafS name="leafs" type="range">
    <infraNodeBlk name="test" from_"105" to_"105"/>
  </infraLeafS>
  <infraRsAccPortP tDn="uni/infra/accportprof-fex116nif105" />
</infraNodeP>

<infraNodeP name="fexNodeP101">
  <infraLeafS name="leafs" type="range">
    <infraNodeBlk name="test" from_"101" to_"101"/>
  </infraLeafS>
  <infraRsAccPortP tDn="uni/infra/accportprof-fex113nif101" />
</infraNodeP>

<infraAccPortP name="fex116nif105">
  <infraHPortS name="pselc" type="range">
  <infraPortBlk name="blk1"
    fromCard="1" toCard="1" fromPort="45" toPort="48" >
  </infraPortBlk>
  <infraRsAccBaseGrp tDn="uni/infra/fexprof-fexHIF116/fexbundle-fex116" fexId="116" />
</infraHPortS>
</infraAccPortP>

<infraAccPortP name="fex113nif101">
  <infraHPortS name="pselc" type="range">
  <infraPortBlk name="blk1"
    fromCard="1" toCard="1" fromPort="45" toPort="48" >
  </infraPortBlk>
  <infraRsAccBaseGrp tDn="uni/infra/fexprof-fexHIF113/fexbundle-fex113" fexId="113" />
</infraHPortS>
</infraAccPortP>

<infraFexP name="fexHIF113">
  <infraFexBndlGrp name="fex113"/>
  <infraHPortS name="pselc-fexPC" type="range">
  <infraPortBlk name="blk"
    fromCard="1" toCard="1" fromPort="15" toPort="16" >
  </infraPortBlk>
  <infraRsAccBaseGrp tDn="uni/infra/funcprof/accbundle-fexPCbundle" />
</infraHPortS>
```

```

    <infraHPortS name="pselc-fexVPC" type="range">
    <infraPortBlk name="blk"
      fromCard="1" toCard="1" fromPort="1" toPort="8" >
    </infraPortBlk>
    <infraRsAccBaseGrp tDn="uni/infra/funcprof/accbundle-fexvpbundle" />
  </infraHPortS>
  <infraHPortS name="pselc-fexaccess" type="range">
    <infraPortBlk name="blk"
      fromCard="1" toCard="1" fromPort="47" toPort="47">
    </infraPortBlk>
    <infraRsAccBaseGrp tDn="uni/infra/funcprof/accportgrp-fexaccport" />
  </infraHPortS>
</infraFexP>

<infraFexP name="fexHIF116">
  <infraFexBndlGrp name="fex116"/>
  <infraHPortS name="pselc-fexPC" type="range">
    <infraPortBlk name="blk"
      fromCard="1" toCard="1" fromPort="17" toPort="18" >
    </infraPortBlk>
    <infraRsAccBaseGrp tDn="uni/infra/funcprof/accbundle-fexPCbundle" />
  </infraHPortS>
  <infraHPortS name="pselc-fexVPC" type="range">
    <infraPortBlk name="blk"
      fromCard="1" toCard="1" fromPort="1" toPort="8" >
    </infraPortBlk>
    <infraRsAccBaseGrp tDn="uni/infra/funcprof/accbundle-fexvpbundle" />
  </infraHPortS>
  <infraHPortS name="pselc-fexaccess" type="range">
    <infraPortBlk name="blk"
      fromCard="1" toCard="1" fromPort="47" toPort="47">
    </infraPortBlk>
    <infraRsAccBaseGrp tDn="uni/infra/funcprof/accportgrp-fexaccport" />
  </infraHPortS>
</infraFexP>

<infraFuncP>
<infraAccBndlGrp name="fexPCbundle" lagT="link">
  <infraRsLacpPol tnLacpLagPolName='staticLag' />
  <infraRsHIfPol tnFabricHIfPolName="1GHIfPol" />
  <infraRsAttEntP tDn="uni/infra/attentp-fexvpcAttEP"/>
</infraAccBndlGrp>

<infraAccBndlGrp name="fexvpbundle" lagT="node">
  <infraRsLacpPol tnLacpLagPolName='staticLag' />
  <infraRsHIfPol tnFabricHIfPolName="1GHIfPol" />
  <infraRsAttEntP tDn="uni/infra/attentp-fexvpcAttEP"/>
</infraAccBndlGrp>
</infraFuncP>

<fabricHIfPol name="1GHIfPol" speed="1G" />
<infraAttEntityP name="fexvpcAttEP">
  <infraProvAcc name="provfunc" />
  <infraRsDomP tDn="uni/phys-fexvpcDOM"/>
</infraAttEntityP>

<lacpLagPol dn="uni/infra/lacplagp-staticLag"
  ctrl="susp-individual,graceful-conv"
  minLinks="2"

```

```
maxLinks="16">  
</lacpLagPol>
```

FCoE

Supporting Fibre Channel over Ethernet Traffic on the ACI Fabric

Cisco ACI enables you to configure and manage support for Fibre Channel over Ethernet (FCoE) traffic on the ACI fabric.

FCoE is a protocol that encapsulates Fibre Channel (FC) packets within Ethernet packets, thus enabling storage traffic to move seamlessly between a Fibre Channel SAN and an Ethernet network.

A typical implementation of FCoE protocol support on the ACI fabric enables hosts located on the Ethernet-based ACI fabric to communicate with SAN storage devices located on an FC network. The hosts are connecting through virtual F ports deployed on an ACI leaf switch. The SAN storage devices and FC network are connected through a Fibre Channel Forwarding (FCF) bridge to the ACI fabric through a virtual NP port, deployed on the same ACI leaf switch as is the virtual F port. Virtual NP ports and virtual F ports are also referred to generically as virtual Fibre Channel (vFC) ports.

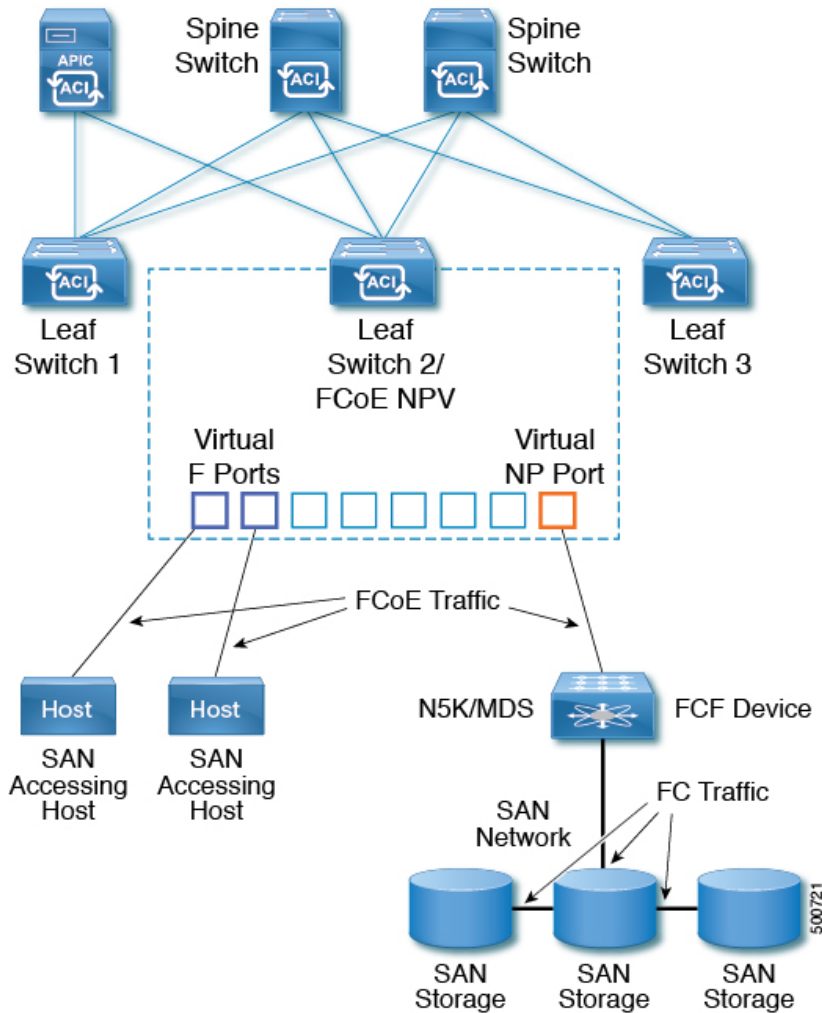


Note In the FCoE topology, the role of the ACI leaf switch is to provide a path for FCoE traffic between the locally connected SAN hosts and a locally connected FCF device. The leaf switch does not perform local switching between SAN hosts, and the FCoE traffic is not forwarded to a spine switch.

Topology Supporting FCoE Traffic Through ACI

The topology of a typical configuration supporting FCoE traffic over the ACI fabric consists of the following components:

Figure 22: ACI Topology Supporting FCoE Traffic



- One or more ACI leaf switches configured through FC SAN policies to function as an NPV backbone.
- Selected interfaces on the NPV-configured leaf switches configured to function as virtual F ports, which accommodate FCoE traffic to and from hosts running SAN management or SAN-consuming applications.
- Selected interfaces on the NPV-configured leaf switches configured to function as virtual NP ports, which accommodate FCoE traffic to and from a Fibre Channel Forwarding (FCF) bridge.

The FCF bridge receives FC traffic from fibre channel links typically connecting SAN storage devices and encapsulates the FC packets into FCoE frames for transmission over the ACI fabric to the SAN management or SAN Data-consuming hosts. It receives FCoE traffic and repackages it back to FC for transmission over the fibre channel network.



Note In the above ACI topology, FCoE traffic support requires direct connections between the hosts and virtual F ports and direct connections between the FCF device and the virtual NP port.

APIC servers enable an operator to configure and monitor the FCoE traffic through the APIC GUI, the APIC NX-OS style CLI, or through application calls to the APIC REST API.

Topology Supporting FCoE Initialization

In order for FCoE traffic flow to take place as described, you must also set up separate VLAN connectivity over which SAN Hosts broadcast FCoE Initialization protocol (FIP) packets to discover the interfaces enabled as F ports.

vFC Interface Configuration Rules

Whether you set up the vFC network and EPG deployment through the APIC GUI, NX-OS style CLI, or the REST API, the following general rules apply across platforms:

- F port mode is the default mode for vFC ports. NP port mode must be specifically configured in the Interface policies.
- The load balancing default mode is for leaf-switch or interface level vFC configuration is src-dst-ox-id.
- One VSAN assignment per bridge domain is supported.
- The allocation mode for VSAN pools and VLAN pools must always be static.
- vFC ports require association with a VSAN domain (also called Fibre Channel domain) that contains VSANs mapped to VLANs.

Configuring FCoE Connectivity Using the REST API

You can configure FCoE-enabled interfaces and EPGs accessing those interfaces using the FCoE protocol with the REST API.

Step 1 To create a VSAN pool, send a post with XML such as the following example.

The example creates VSAN pool **vsanPool1** and specifies the range of VSANs to be included.

Example:

```
https://apic-ip-address/api/mo/uni/infra/vsanns-[vsanPool1]-static.xml

<!-- Vsan-pool -->
<fvnsVsanInstP name="vsanPool1" allocMode="static">
  <fvnsVsanEncapBlk name="encap" from="vsan-5" to="vsan-100"/>
</fvnsVsanInstP>
```

Step 2 To create a VLAN pool, send a post with XML such as the following example.

The example creates VLAN pool **vlanPool1** and specifies the range of VLANs to be included.

Example:

```
https://apic-ip-address/api/mo/uni/infra/vlanns-[vlanPool1]-static.xml

<!-- Vlan-pool -->
<fvnsVlanInstP name="vlanPool1" allocMode="static">
  <fvnsEncapBlk name="encap" from="vlan-5" to="vlan-100"/>
</fvnsVlanInstP>
```

Step 3 To create a VSAN-Attribute policy, send a post with XML such as the following example.

The example creates VSAN attribute policy **vsanattri1**, maps **vsan-10** to **vlan-43**, and maps **vsan-11** to **vlan-44**.

Example:

```
https://apic-ip-address/api/mo/uni/infra/vsanattrp-[vsanattr1].xml

<fcVsanAttrP name="vsanattr1">

    <fcVsanAttrPEntry vlanEncap="vlan-43" vsanEncap="vsan-10"/>
    <fcVsanAttrPEntry vlanEncap="vlan-44" vsanEncap="vsan-11"
        lbType="src-dst-ox-id"/>
</fcVsanAttrP>
```

Step 4 To create a Fibre Channel domain, send a post with XML such as the following example.

The example creates VSAN domain **vsanDom1**.

Example:

```
https://apic-ip-address/api/mo/uni/fc-vsanDom1.xml
<!-- Vsan-domain -->
<fcDomP name="vsanDom1">
    <fcRsVsanAttr tDn="uni/infra/vsanattrp-[vsanattr1]"/>
    <infraRsVlanNs tDn="uni/infra/vlanns-[vlanPool1]-static"/>
    <fcRsVsanNs tDn="uni/infra/vsanns-[vsanPool1]-static"/>
</fcDomP>
```

Step 5 To create the tenant, application profile, EPG and associate the FCoE bridge domain with the EPG, send a post with XML such as the following example.

The example creates a bridge domain **bd1** under a target tenant configured to support FCoE and an application EPG **epg1**. It associates the EPG with VSAN domain **vsanDom1** and a Fibre Channel path (to interface **1/39** on leaf switch **101**. It deletes a Fibre channel path to interface **1/40** by assigning the `<fvRsFcPathAtt>` object with "deleted" status. Each interface is associated with a VSAN.

Note Two other possible alternative vFC deployments are also displayed. One sample deploys vFC on a port channel. The other sample deploys vFC on a virtual port channel.

Example:

```
https://apic-ip-address/api/mo/uni/tn-tenant1.xml

<fvTenant
name="tenant1">
    <fvCtx name="vrf1"/>

    <!-- bridge domain -->
    <fvBD name="bd1" type="fc" >
        <fvRsCtx tnFvCtxName="vrf1" />
    </fvBD>

    <fvAp name="app1">
        <fvAEPg name="epg1">
            <fvRsBd tnFvBDName="bd1" />
            <fvRsDomAtt tDn="uni/fc-vsanDom1" />
            <fvRsFcPathAtt tDn="topology/pod-1/paths-101/pathep-[eth1/39]"
                vsan="vsan-11" vsanMode="native"/>
            <fvRsFcPathAtt tDn="topology/pod-1/paths-101/pathep-[eth1/40]"
                vsan="vsan-10" vsanMode="regular" status="deleted"/>
        </fvAEPg>

    <!-- Sample deployment of vFC on a port channel -->

        <fvRsFcPathAtt vsanMode="native" vsan="vsan-10"
```



```

    tDn="topology/pod-1/paths 101/pathep-pc01"/>
<!-- Sample deployment of vFC on a virtual port channel -->
    <fvRsFcPathAtt vsanMode="native" vsan="vsan-10"
      tDn="topology/pod-1/paths-101/pathep-vpc01"/>
    <fvRsFcPathAtt vsanMode="native" vsan="vsan-10"
      tDn="topology/pod-1/paths-102/pathep-vpc01"/>
  </fvAp>
</fvTenant>

```

Step 6 To create a port policy group and an AEP, send a post with XML such as the following example.

The example executes the following requests:

- Creates a policy group **portgrp1** that includes an FC interface policy **fcIfPol1**, a priority flow control policy **pfcIfPol1** and a slow-drain policy **sdIfPol1**.
- Creates an attached entity profile (AEP) **AttEntP1** that associates the ports in VSAN domain **vsanDom1** with the settings to be specified for **fcIfPol1**, **pfcIfPol1**, and **sdIfPol1**.

Example:

`https://apic-ip-address/api/mo/uni.xml`

```

<polUni>
  <infraInfra>
    <infraFuncP>
      <infraAccPortGrp name="portgrp1">
        <infraRsFcIfPol tnFcIfPolName="fcIfPol1"/>
        <infraRsAttEntP tDn="uni/infra/attentp-AttEntP1" />
        <infraRsQosPfcIfPol tnQosPfcIfPolName="pfcIfPol1"/>
        <infraRsQosSdIfPol tnQosSdIfPolName="sdIfPol1"/>
      </infraAccPortGrp>
    </infraFuncP>

    <infraAttEntityP name="AttEntP1">
      <infraRsDomP tDn="uni/fc-vsanDom1"/>
    </infraAttEntityP>
    <qosPfcIfPol dn="uni/infra/pfc-pfcIfPol1" adminSt="on">
    </qosPfcIfPol>
    <qosSdIfPol dn="uni/infra/qosdpol-sdIfPol1" congClearAction="log"
      congDetectMult="5" flushIntvl="100" flushAdminSt="enabled">
    </qosSdIfPol>
    <fcIfPol dn="uni/infra/fcIfPol-fcIfPol1" portMode="np">
    </fcIfPol>

  </infraInfra>
</polUni>

```

Step 7 To create a node selector and a port selector, send a post with XML such as the following example.

The example executes the following requests:

- Creates node selector **leafsel1** that specifies leaf node **101**.
- Creates port selector **portsel1** that specifies port **1/39**.

Example:

```

https://apic-ip-address/api/mo/uni.xml

<polUni>
  <infraInfra>
    <infraNodeP name="nprof1">
      <infraLeafS name="leafsell" type="range">
        <infraNodeBlk name="nblk1" from_="101" to_="101"/>
      </infraLeafS>
      <infraRsAccPortP tDn="uni/infra/accportprof-pprof1"/>
    </infraNodeP>

    <infraAccPortP name="pprof1">
      <infraHPortS name="portsell" type="range">
        <infraPortBlk name="blk"
          fromCard="1" toCard="1" fromPort="39" toPort="39">
        </infraPortBlk>

        <infraRsAccBaseGrp tDn="uni/infra/funcprof/accportgrp-portgrp1" />
      </infraHPortS>

    </infraAccPortP>
  </infraInfra>
</polUni>

```

Step 8 To create a vPC, send a post with XML such as the following example.

Example:

```

https://apic-ip-address/api/mo/uni.xml
<polUni>
  <fabricInst>

    <vpcInstPol name="vpc01" />

    <fabricProtPol pairT="explicit" >
      <fabricExplicitGEp name="vpc01" id="100" >
        <fabricNodePEp id="101"/>
        <fabricNodePEp id="102"/>
        <fabricRsVpcInstPol tnVpcInstPolName="vpc01" />
        <!-- <fabricLagId accBndlGrp="infraAccBndlGrp_{pcname}" /> -->
      </fabricExplicitGEp>
    </fabricProtPol>

  </fabricInst>
</polUni>

```

Configuring FCoE Over FEX Using REST API

Before you begin

- Follow the steps 1 through 4 as described in [Configuring FCoE Connectivity Using the REST API](#), on page 245

Step 1 Configure FCoE over FEX (Selectors): Port:

Example:

```

<infraInfra dn="uni/infra">
  <infraNodeP name="nprof1">
    <infraLeafS name="leafsell" type="range">
      <infraNodeBlk name="nblk1" from_="101" to_="101"/>
    </infraLeafS>
    <infraRsAccPortP tDn="uni/infra/accportprof-pprof1" />
  </infraNodeP>

  <infraAccPortP name="pprof1">
    <infraHPortS name="portsell" type="range">
      <infraPortBlk name="blk"
        fromCard="1" toCard="1" fromPort="17" toPort="17"></infraPortBlk>
      <infraRsAccBaseGrp tDn="uni/infra/fexprof-fexprof1/fexbundle-fexbundle1" fexId="110" />
    </infraHPortS>
  </infraAccPortP>

  <infraFuncP>
    <infraAccPortGrp name="portgrp1">
      <infraRsAttEntP tDn="uni/infra/attentp-attentp1" />
    </infraAccPortGrp>
  </infraFuncP>

  <infraFexP name="fexprof1">
    <infraFexBndlGrp name="fexbundle1"/>
    <infraHPortS name="portsel2" type="range">
      <infraPortBlk name="blk2"
        fromCard="1" toCard="1" fromPort="20" toPort="20"></infraPortBlk>
      <infraRsAccBaseGrp tDn="uni/infra/funcprof/accportgrp-portgrp1"/>
    </infraHPortS>
  </infraFexP>

  <infraAttEntityP name="attentp1">
    <infraRsDomP tDn="uni/fc-vsanDom1"/>
  </infraAttEntityP>
</infraInfra>

```

Step 2 Tenant configuration:**Example:**

```

fvTenant name="tenant1">
<fvCtx name="vrf1"/>

<!-- bridge domain -->
  <fvBD name="bd1" type="fc" >
    <fvRsCtx tnFvCtxName="vrf1" />
  </fvBD>

  <fvAp name="ap1">
    <fvAEPg name="epg1">
      <fvRsBd tnFvBDName="bd1" />
      <fvRsDomAtt tDn="uni/fc-vsanDom1" />
    <fvRsFcPathAtt tDn="topology/pod-1/paths-101/extpaths-110/paths-110/epg-[eth1/17]" vsan="vsan-11"
      vsanMode="native"/>
    </fvAEPg>
  </fvAp>
</fvTenant>

```

Step 3 Configure FCoE over FEX (Selectors): Port-Channel:**Example:**

```

<infraInfra dn="uni/infra">
  <infraNodeP name="nprof1">
    <infraLeafS name="leafsell" type="range">

```

```

    <infraNodeBlk name="nblk1" from_="101" to_="101"/>
  </infraLeafS>
  <infraRsAccPortP tDn="uni/infra/accportprof-pprofl" />
</infraNodeP>

  <infraAccPortP name="pprofl">
    <infraHPortS name="portsell" type="range">
      <infraPortBlk name="blk1"
        fromCard="1" toCard="1" fromPort="18" toPort="18"></infraPortBlk>
      <infraRsAccBaseGrp tDn="uni/infra/fexprof-fexprofl/fexbundle-fexbundle1" fexId="111" />
    </infraHPortS>
  </infraAccPortP>

  <infraFexP name="fexprofl">
    <infraFexBndlGrp name="fexbundle1"/>
    <infraHPortS name="portsell1" type="range">
      <infraPortBlk name="blk1"
        fromCard="1" toCard="1" fromPort="20" toPort="20"></infraPortBlk>
      <infraRsAccBaseGrp tDn="uni/infra/funcprof/accbundle-pc1"/>
    </infraHPortS>
  </infraFexP>

  <infraFuncP>
    <infraAccBndlGrp name="pc1">
      <infraRsAttEntP tDn="uni/infra/attentp-attentp1" />
    </infraAccBndlGrp>
  </infraFuncP>

  <infraAttEntityP name="attentp1">
<infraRsDomP tDn="uni/fc-vsanDom1"/>
  </infraAttEntityP>
</infraInfra>

```

Step 4 Tenant configuration:**Example:**

```

<fvTenant name="tenant1">
  <fvCtx name="vrfl1"/>

  <!-- bridge domain -->
  <fvBD name="bd1" type="fc" >
    <fvRsCtx tnFvCtxName="vrfl1" />
  </fvBD>

  <fvAp name="appl1">
    <fvAEPg name="epg1">
      <fvRsBd tnFvBDName="bd1" />
      <fvRsDomAtt tDn="uni/fc-vsanDom1" />
    <fvRsFcPathAtt tDn="topology/pod-1/paths-101/extpaths-111/pathep-[pc1]" vsan="vsan-11" vsanMode="native"
    />
  </fvAEPg>
  </fvAp>
</fvTenant>

```

Step 5 Configure FCoE over FEX (Selectors): vPC:**Example:**

```

<polUni>
  <fabricInst>
  <vpcInstPol name="vpc1" />
  <fabricProtPol pairT="explicit" >
  <fabricExplicitGEp name="vpc1" id="100" >
  <fabricNodePEp id="101"/>

```

```

<fabricNodePEp id="102"/>
<fabricRsVpcInstPol tnVpcInstPolName="vpc1" />
</fabricExplicitGep>
</fabricProtPol>
</fabricInst>
</polUni>

```

Step 6 Tenant configuration:**Example:**

```

<fvTenant name="tenant1">
<fvCtx name="vrf1"/>

<!-- bridge domain -->
  <fvBD name="bd1" type="fc" >
    <fvRsCtx tnFvCtxName="vrf1" />
  </fvBD>

  <fvAp name="app1">
    <fvAEPg name="epg1">
      <fvRsBd tnFvBDName="bd1" />
      <fvRsDomAtt tDn="uni/fc-vsanDom1" />
    <fvRsFcPathAtt vsanMode="native" vsan="vsan-11"
tDn="topology/pod-1/protpaths-101-102/extprotpaths-111-111/pathep-[vpc1]" />
  </fvAEPg>
  </fvAp>
</fvTenant>

```

Step 7 Selector configuration:**Example:**

```

<polUni>
<infraInfra>
<infraNodeP name="nprof1">
<infraLeafS name="leafsel1" type="range">
<infraNodeBlk name="nblk1" from_="101" to_="101"/>
</infraLeafS>
<infraRsAccPortP tDn="uni/infra/accportprof-pprof1" />
</infraNodeP>

<infraNodeP name="nprof2">
<infraLeafS name="leafsel2" type="range">
<infraNodeBlk name="nblk2" from_="102" to_="102"/>
</infraLeafS>
<infraRsAccPortP tDn="uni/infra/accportprof-pprof2" />
</infraNodeP>

<infraAccPortP name="pprof1">
<infraHPortS name="portsel1" type="range">
<infraPortBlk name="blk1"
fromCard="1" toCard="1" fromPort="18" toPort="18">
</infraPortBlk>
<infraRsAccBaseGrp tDn="uni/infra/fexprof-fexprof1/fexbundle-fexbundle1" fexId="111" />
</infraHPortS>
</infraAccPortP>
<infraAccPortP name="pprof2">
<infraHPortS name="portsel2" type="range">
<infraPortBlk name="blk2"
fromCard="1" toCard="1" fromPort="18" toPort="18">
</infraPortBlk>
<infraRsAccBaseGrp tDn="uni/infra/fexprof-fexprof2/fexbundle-fexbundle2" fexId="111" />
</infraHPortS>
</infraAccPortP>

```

```

<infraFexP name="fexprof1">
<infraFexBndlGrp name="fexbundle1"/>
<infraHPortS name="portsel1" type="range">
<infraPortBlk name="blk1"
fromCard="1" toCard="1" fromPort="20" toPort="20">
</infraPortBlk>
  <infraRsAccBaseGrp tDn="uni/infra/funcprof/accbundle-vpc1"/>
</infraHPortS>
</infraFexP>

<infraFexP name="fexprof2">
<infraFexBndlGrp name="fexbundle2"/>
<infraHPortS name="portsel2" type="range">
<infraPortBlk name="blk2"
fromCard="1" toCard="1" fromPort="20" toPort="20">

</infraPortBlk>
<infraRsAccBaseGrp tDn="uni/infra/funcprof/accbundle-vpc1"/>
</infraHPortS>
</infraFexP>

<infraFuncP>
<infraAccBndlGrp name="vpc1" lagT="node">
  <infraRsAttEntP tDn="uni/infra/attentp-attentp1" />
</infraAccBndlGrp>
</infraFuncP>

<infraAttEntityP name="attentp1">
<infraRsDomP tDn="uni/fc-vsanDom1"/>
</infraAttEntityP>
</infraInfra>
</polUni>

```

Undeploying FCoE Connectivity through the REST API or SDK

To undeploy FCoE connectivity through the APIC REST API or SDK, delete the following objects associated with the deployment:

Object	Description
<fvRsFcPathAtt> (Fibre Channel Path)	The Fibre Channel path specifies the vFC path to the actual interface. Deleting each object of this type removes the deployment from that object's associated interfaces.
<fcVsanAttrp> (VSAN/VLAN map)	The VSAN/VLAN map maps the VSANs to their associated VLANs deleting this object removes the association between the VSANs that support FCoE connectivity and their underlying VSANs.
<fvnsVsanInstP> (VSAN pool)	The VSAN pool specifies the set of VSANs available to support FCoE connectivity. Deleting this pool removes those VSANs.

Object	Description
<fvnsVlanIsntP> ((VLAN pool)	The VLAN pool specifies the set of VLANs available for VSAN mapping. Deleting the associated VLAN pool cleans up after an FCoE undeployment, removing the underlying VLAN entities over which the VSAN entities ran.
<fcDomP> (VSAN or Fibre Channel domain)	The Fibre Channel domain includes all the VSANs and their mappings. Deleting this object undeploys vFC from all interfaces associated with this domain.
<fvAEPg> (application EPG)	The application EPG associated with the FCoE connectivity. If the purpose of the application EPGs was only to support FCoE-related activity, you might consider deleting this object.
<fvAp> (application profile)	The application profile associated with the FCoE connectivity. If the purpose of the application profile was only to support FCoE-related activity, you might consider deleting this object.
<fvTenant> (tenant)	The tenant associated with the FCoE connectivity. If the purpose of the tenant was only to support FCoE-related activity, you might consider deleting this object.



Note If during clean up you delete the Ethernet configuration object (infraHPortS) for a vFC port, the default vFC properties remain associated with that interface. For example if the interface configuration for vFC NP port 1/20 is deleted, that port remains a vFC port but with default F port setting rather than non-default NP port setting applied.

The following steps undeploy FCoE-enabled interfaces and EPGs accessing those interfaces using the FCoE protocol.

Step 1

To delete the associated Fibre Channel path objects, send a post with XML such as the following example.

The example deletes all instances of the Fibre Channel path object <fvRsFcPathAtt>.

Note Deleting the Fibre Channel paths undeploys the vFC from the ports/VSANs that used them.

Example:

`https://apic-ip-address/api/mo/uni/tn-tenant1.xml`

```
<fvTenant
name="tenant1">
  <fvCtx name="vrf1"/>

  <!-- bridge domain -->
  <fvBD name="bd1" type="fc" >
    <fvRsCtx tnFvCtxName="vrf1" />
  </fvBD>

  <fvAp name="app1">
    <fvAEPg name="epg1">
      <fvRsBd tnFvBDName="bd1" />
    </fvAEPg>
  </fvAp>
</fvTenant>
```

```

    <fvRsDomAtt tDn="uni/fc-vsanDom1" />
    <fvRsFcPathAtt tDn="topology/pod-1/paths-101/pathep-[eth1/39]"
      vsan="vsan-11" vsanMode="native" status="deleted"/>
    <fvRsFcPathAtt tDn="topology/pod-1/paths-101/pathep-[eth1/40]"
      vsan="vsan-10" vsanMode="regular" status="deleted"/>
  </fvAEPg>

<!-- Sample undeployment of vFC on a port channel -->

    <fvRsFcPathAtt vsanMode="native" vsan="vsan-10"
      tDN="topology/pod-1/paths 101/pathep-pc01" status="deleted"/>

<!-- Sample undeployment of vFC on a virtual port channel -->

    <fvRsFcPathAtt vsanMode="native" vsan="vsan-10"
      tDn="topology/pod-1/paths-101/pathep-vpc01" status="deleted"/>
    <fvRsFcPathAtt vsanMode="native" vsan="vsan-10"
      tDn="topology/pod-1/paths-102/pathep-vpc01" status="deleted"/>

  </fvAp>
</fvTenant>

```

- Step 2** To delete the associated VSAN/VLAN map, send a post such as the following example. The example deletes the VSAN/VLAN map **vsanattr1** and its associated `<fcVsanAttrP>` object.

Example:

```
https://apic-ip-address/api/mo/uni/infra/vsanattrp-[vsanattr1].xml
```

```

<fcVsanAttrP name="vsanattr1" status="deleted">
  <fcVsanAttrPEntry vlanEncap="vlan-43" vsanEncap="vsan-10" status="deleted"/>
  <fcVsanAttrPEntry vlanEncap="vlan-44" vsanEncap="vsan-11"
    lbType="src-dst-ox-id" status="deleted" />
</fcVsanAttrP>

```

- Step 3** To delete the associated VSAN pool, send a post such as the following example. The example deletes the VSAN pool **vsanPool1** and its associated `<fvnsVsanInstP>` object.

Example:

```
https://apic-ip-address/api/mo/uni/infra/vsanns-[vsanPool1]-static.xml
```

```

<!-- Vsan-pool -->
<fvnsVsanInstP name="vsanPool1" allocMode="static" status="deleted">
  <fvnsVsanEncapBlk name="encap" from="vsan-5" to="vsan-100" />
</fvnsVsanInstP>

```

- Step 4** To delete the associated VLAN pool, send a post with XML such as the following example. The example deletes the VLAN pool **vlanPool1** and its associated `<fvnsVlanInstP>` object.

Example:

```
https://apic-ip-address/api/mo/uni/infra/vlanns-[vlanPool1]-static.xml
```

```

<!-- Vlan-pool -->
<fvnsVlanInstP name="vlanPool1" allocMode="static" status="deleted">
  <fvnsEncapBlk name="encap" from="vlan-5" to="vlan-100" />
</fvnsVlanInstP>

```

- Step 5** To delete the associated Fibre Channel domain, send a post with XML such as the following example. The example deletes the VSAN domain **vsanDom1** and its associated `<fcDomP>` object.

Example:

```
https://apic-ip-address/api/mo/uni/fc-vsanDom1.xml
<!-- Vsan-domain -->
<fcDomP name="vsanDom1" status="deleted">
  <fcRsVsanAttr tDn="uni/infra/vsanattrp-[vsanattr1]"/>
  <infraRsVlanNs tDn="uni/infra/vlanns-[vlanPool1]-static"/>
  <fcRsVsanNs tDn="uni/infra/vsanns-[vsanPool1]-static"/>
</fcDomP>
```

Step 6 **Optional:** If appropriate, you can delete the associated application EPG, the associated application profile, or the associated tenant.

Example:

In the following sample, the associated application EPG **epg1** and its associated <fvAEPg> object is deleted.

```
https://apic-ip-address/api/mo/uni/tn-tenant1.xml

<fvTenant
name="tenant1"/>
  <fvCtx name="vrf1"/>

  <!-- bridge domain -->
  <fvBD name="bd1" type="fc" >
    <fvRsCtx tnFvCtxName="vrf1" />
  </fvBD>

  <fvAp name="app1">
    <fvAEPg name="epg1" status="deleted">
      <fvRsBd tnFvBDName="bd1" />
      <fvRsDomAtt tDn="uni/fc-vsanDom1" />
      <fvRsFcPathAtt tDn="topology/pod-1/paths-101/pathep-[eth1/39]"
        vsan="vsan-11" vsanMode="native" status="deleted"/>
      <fvRsFcPathAtt tDn="topology/pod-1/paths-101/pathep-[eth1/40]"
        vsan="vsan-10" vsanMode="regular" status="deleted"/>
    </fvAEPg>

    <!-- Sample undeployment of vFC on a port channel -->

    <fvRsFcPathAtt vsanMode="native" vsan="vsan-10"
      tDn="topology/pod-1/paths 101/pathep-pc01" status="deleted"/>

    <!-- Sample undeployment of vFC on a virtual port channel -->

    <fvRsFcPathAtt vsanMode="native" vsan="vsan-10"
      tDn="topology/pod-1/paths-101/pathep-vpc01" status="deleted"/>
    <fvRsFcPathAtt vsanMode="native" vsan="vsan-10"
      tDn="topology/pod-1/paths-102/pathep-vpc01" status="deleted"/>

  </fvAp>
</fvTenant>
```

Example:

In the following example, the associated application profile **app1** and its associated <fvAp> object is deleted.

```
https://apic-ip-address/api/mo/uni/tn-tenant1.xml

<fvTenant
name="tenant1">
  <fvCtx name="vrf1"/>

  <!-- bridge domain -->
  <fvBD name="bd1" type="fc">
    <fvRsCtx tnFvCtxName="vrf1" />
```

```

</fvBD>

<fvAp name="appl" status="deleted">
  <fvAEPg name="epg1" status="deleted">
    <fvRsBd tnFvBDName="bd1" />
    <fvRsDomAtt tDn="uni/fc-vsanDom1" />
    <fvRsFcPathAtt tDn="topology/pod-1/paths-101/pathep-[eth1/39]"
      vsan="vsan-11" vsanMode="native" status="deleted"/>
    <fvRsFcPathAtt tDn="topology/pod-1/paths-101/pathep-[eth1/40]"
      vsan="vsan-10" vsanMode="regular" status="deleted"/>
  </fvAEPg>

<!-- Sample undeployment of vFC on a port channel -->

  <fvRsFcPathAtt vsanMode="native" vsan="vsan-10"
    tDN="topology/pod-1/paths 101/pathep-pc01" status="deleted"/>

<!-- Sample undeployment of vFC on a virtual port channel -->

  <fvRsFcPathAtt vsanMode="native" vsan="vsan-10"
    tDn="topology/pod-1/paths-101/pathep-vpc01" status="deleted"/>
  <fvRsFcPathAtt vsanMode="native" vsan="vsan-10"
    tDn="topology/pod-1/paths-102/pathep-vpc01" status="deleted"/>

</fvAp>
</fvTenant>

```

Example:

In the following example, the entire tenant **tenant1** and its associated `<fvTenant>` object is deleted.

`https://apic-ip-address/api/mo/uni/tn-tenant1.xml`

```

<fvTenant
name="tenant1" status="deleted">
  <fvCtx name="vrf1"/>

  <!-- bridge domain -->
  <fvBD name="bd1" type="fc" status="deleted">
    <fvRsCtx tnFvCtxName="vrf1" />
  </fvBD>

  <fvAp name="appl">
    <fvAEPg name="epg1" status="deleted">
      <fvRsBd tnFvBDName="bd1" />
      <fvRsDomAtt tDn="uni/fc-vsanDom1" />
      <fvRsFcPathAtt tDn="topology/pod-1/paths-101/pathep-[eth1/39]"
        vsan="vsan-11" vsanMode="native" status="deleted"/>
      <fvRsFcPathAtt tDn="topology/pod-1/paths-101/pathep-[eth1/40]"
        vsan="vsan-10" vsanMode="regular" status="deleted"/>
    </fvAEPg>

  <!-- Sample undeployment of vFC on a port channel -->

    <fvRsFcPathAtt vsanMode="native" vsan="vsan-10"
      tDN="topology/pod-1/paths 101/pathep-pc01" status="deleted"/>

  <!-- Sample undeployment of vFC on a virtual port channel -->

    <fvRsFcPathAtt vsanMode="native" vsan="vsan-10"
      tDn="topology/pod-1/paths-101/pathep-vpc01" status="deleted"/>
    <fvRsFcPathAtt vsanMode="native" vsan="vsan-10"
      tDn="topology/pod-1/paths-102/pathep-vpc01" status="deleted"/>

```

```
</fvAp>  
</fvTenant>
```

Fibre Channel NPV

Fibre Channel Connectivity Overview

Cisco ACI supports Fibre Channel (FC) connectivity on a leaf switch using N-Port Virtualization (NPV) mode. NPV allows the switch to aggregate FC traffic from locally connected host ports (N ports) into a node proxy (NP port) uplink to a core switch.

A switch is in NPV mode after enabling NPV. NPV mode applies to an entire switch. Each end device connected to an NPV mode switch must log in as an N port to use this feature (loop-attached devices are not supported). All links from the edge switches (in NPV mode) to the NPV core switches are established as NP ports (not E ports), which are used for typical inter-switch links.



Note In the FC NPV application, the role of the ACI leaf switch is to provide a path for FC traffic between the locally connected SAN hosts and a locally connected core switch. The leaf switch does not perform local switching between SAN hosts, and the FC traffic is not forwarded to a spine switch.

FC NPV Benefits

FC NPV provides the following:

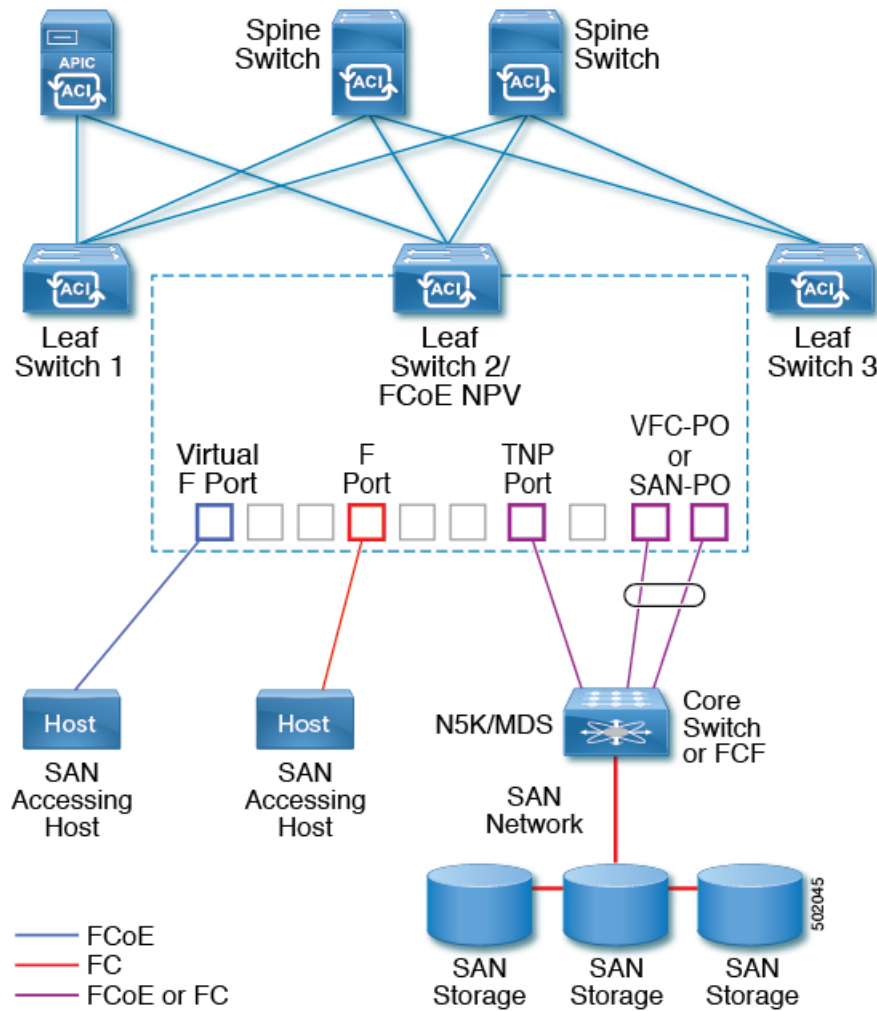
- Increases the number of hosts that connect to the fabric without adding domain IDs in the fabric. The domain ID of the NPV core switch is shared among multiple NPV switches.
- FC and FCoE hosts connect to SAN fabrics using native FC interfaces.
- Automatic traffic mapping for load balancing. For newly added servers connected to NPV, traffic is automatically distributed among the external uplinks based on current traffic loads.
- Static traffic mapping. A server connected to NPV can be statically mapped to an external uplink.

FC NPV Mode

Feature-set `fcoe-npv` in ACI will be enabled automatically by default when the first FCoE/FC configuration is pushed.

FC Topology

The topology of various configurations supporting FC traffic over the ACI fabric is shown in the following figure:



- Server/storage host interfaces on the ACI leaf switch can be configured to function as either native F ports or as virtual FC (FCoE) ports.
- An uplink interface to a FC core switch can be configured as any of the following port types:
 - native FC NP port
 - SAN-PO NP port
- An uplink interface to a FCF switch can be configured as any of the following port types:
 - virtual (vFC) NP port
 - vFC-PO NP port
- N-Port ID Virtualization (NPIV) is supported and enabled by default, allowing an N port to be assigned multiple N port IDs or Fibre Channel IDs (FCID) over a single link.
- Trunking can be enabled on an NP port to the core switch. Trunking allows a port to support more than one VSAN. When trunk mode is enabled on an NP port, it is referred to as a TNP port.

- Multiple NP ports can be combined as a SAN port channel (SAN-PO) to the core switch. Trunking is supported on a SAN port channel.
- FC F ports support 4/16/32 Gbps and auto speed configuration, but 8Gbps is not supported for host interfaces. The default speed is "auto."
- FC NP ports support 4/8/16/32 Gbps and auto speed configuration. The default speed is "auto."
- Multiple FDISC followed by Flogi (nested NPIV) is supported with FC/FCoE host and FC/FCoE NP links.
- Starting in the 4.1(1) release, an FCoE host behind a FEX is supported over the Fibre Channel NP/uplink.
- All FCoE hosts behind one FEX can either be load balanced across multiple vFC and vFC-PO uplinks, or through a single Fibre Channel/SAN port channel uplink.
- SAN boot is supported on a FEX through a Fibre Channel or SAN port channel uplink.
- Starting in the 4.1(1) release, SAN boot is supported over both FC and FCoE uplinks.

Fibre Channel N-Port Virtualization Guidelines and Limitations

When configuring Fibre Channel N-Port Virtualization (NPV), note the following guidelines and limitations:

- Fibre Channel NP ports support trunk mode, but Fibre Channel F ports do not.
- On a trunk Fibre Channel port, internal login happens on the highest VSAN.
- On the core switch, the following features must be enabled:

```
feature npiv
feature fport-channel-trunk
```

- To use an 8G uplink speed, you must configure the IDLE fill pattern on the core switch.



Note Following is an example of configuring IDLE fill pattern on a Cisco MDS switch:

```
Switch(config)# int fc2/3
Switch(config)# switchport fill-pattern IDLE speed 8000
Switch(config)# show run int fc2/3

interface fc2/3
switchport speed 8000
switchport mode NP
switchport fill-pattern IDLE speed 8000
no shutdown
```

- Fibre Channel NPV support is limited to the Cisco N9K-C93180YC-FX switch.
- You can use ports 1 through 48 for Fibre Channel configuration. Ports 49 through 54 cannot be Fibre Channel ports.

- If you convert a port from Ethernet to Fibre Channel or the other way around, you must reload the switch. Currently, you can convert only one contiguous range of ports to Fibre Channel ports, and this range must be a multiple of 4, ending with a port number that is a multiple of 4. For example, 1-4, 1-8, or 21-24.
- Fibre Channel Uplink (NP) connectivity to Brocade Port Blade Fibre Channel 16-32 is not supported when a Cisco N9K-93180YC-FX leaf switch port is configured in 8G speed.
- The selected port speed must be supported by the SFP. For example, because a 32G SFP supports 8/16/32G, a 4G port speed requires an 8G or 16G SFP. Because a 16G SFP supports 4/8/16G, a 32G port speed requires a 32G SFP.
- Speed autonegotiation is supported. The default speed is 'auto'.
- You cannot use Fibre Channel on 40G and breakout ports.
- FEX cannot be directly connected to FC ports.
- FEX HIF ports cannot be converted to FC.
- SAN boot is supported on FEX for FCoE hosts (not Fibre Channel hosts), but not through a vPC.
- Reloading a switch after changing a switch's port profile configuration interrupts traffic through the data plane.

Configuring FC Connectivity Using the REST API

You can configure FC-enabled interfaces and EPGs accessing those interfaces using the FC protocol with the REST API.

Step 1 To create a VSAN pool, send a post with XML such as the following example. The example creates VSAN pool myVsanPool1 and specifies the range of VSANs to be included as vsan-50 to vsan-60:

Example:

```
https://apic-ip-address/api/mo/uni/infra/vsanns-[myVsanPool1]-static.xml

<fvnsVsanInstP allocMode="static" name="myVsanPool1">
  <fvnsVsanEncapBlk from="vsan-50" name="encap" to="vsan-60"/>
</fvnsVsanInstP>
```

Step 2 To create a Fibre Channel domain, send a post with XML such as the following example. The example creates Fibre Channel domain (VSAN domain) myFcDomain1 and associates it with the VSAN pool myVsanPool1:

Example:

```
https://apic-ip-address/api/mo/uni/fc-myFcDomain1.xml

<fcDomP name="myFcDomain1">
  <fcRsVsanNs tDn="uni/infra/vsanns-[myVsanPool1]-static"/>
</fcDomP>
```

Step 3 To create an Attached Entity Policy (AEP) for the FC ports, send a post with XML such as the following example. The example creates the AEP myFcAEP1 and associates it with the Fibre Channel domain myFcDomain1:

Example:

```

https://apic-ip-address/api/mo/uni.xml

<polUni>
<infraInfra>
  <infraAttEntityP name="myFcAEP1">
    <infraRsDomP tDn="uni/fc-myFcDomain1"/>
  </infraAttEntityP>
</infraInfra>
</polUni>

```

Step 4 To create a FC interface policy and a policy group for server host ports, send a post with XML. This example executes the following requests:

- Creates a FC interface policy myFcHostIfPolicy1 for server host ports. These are F ports with no trunking.
- Creates a FC interface policy group myFcHostPortGroup1 that includes the FC host interface policy myFcHostIfPolicy1.
- Associates the policy group to the FC interface policy to convert these ports to FC ports.
- Creates a host port profile myFcHostPortProfile.
- Creates a port selector myFcHostSelector that specifies ports in range 1/1-8.
- Creates a node selector myFcNode1 that specifies leaf node 104.
- Creates a node selector myLeafSelector that specifies leaf node 104.
- Associates the host ports to the leaf node.

Example:

```

https://apic-ip-address/api/mo/uni.xml

<polUni>
  <infraInfra>
    <fcIfPol name="myFcHostIfPolicy1" portMode="f" trunkMode="trunk-off" speed="auto"/>
    <infraFuncP>
      <infraFcAccPortGrp name="myFcHostPortGroup1">
        <infraRsFcL2IfPol tnFcIfPolName="myFcHostIfPolicy1" />
      </infraFcAccPortGrp>
    </infraFuncP>
    <infraAccPortP name="myFcHostPortProfile">
      <infraHPortS name="myFcHostSelector" type="range">
        <infraPortBlk name="myHostPorts" fromCard="1" toCard="1" fromPort="1" toPort="8" />
        <infraRsAccBaseGrp tDn="uni/infra/funcprof/fcaccportgrp-myFcHostPortGroup1" />
      </infraHPortS>
    </infraAccPortP>
    <infraNodeP name="myFcNode1">
      <infraLeafS name="myLeafSelector" type="range">
        <infraNodeBlk name="myLeaf104" from_"104" to_"104" />
      </infraLeafS>
      <infraRsAccPortP tDn="uni/infra/accportprof-myHostPorts" />
    </infraNodeP>
  </infraInfra>
</polUni>

```

Note When this configuration is applied, a switch reload is required to bring up the ports as FC ports.

Currently only one contiguous range of ports can be converted to FC ports, and this range must be multiple of 4 ending with a port number that is multiple of 4. Examples are 1-4, 1-8, or 21-24.

Step 5 To create a FC uplink port interface policy and a policy group for uplink port channels, send a post with XML. This example executes the following requests:

- Creates a FC interface policy myFcUplinkIfPolicy2 for uplink ports. These are NP ports with trunking enabled.
- Creates a FC interface bundle policy group myFcUplinkBundleGroup2 that includes the FC uplink interface policy myFcUplinkIfPolicy2.
- Associates the policy group to the FC interface policy to convert these ports to FC ports.
- Creates an uplink port profile myFcUplinkPortProfile.
- Creates a port selector myFcUplinkSelector that specifies ports in range 1/9-12.
- Associates the host ports to the leaf node 104.

Example:

`https://apic-ip-address/api/mo/uni.xml`

```
<polUni>
  <infraInfra>
    <fcIfPol name="myFcUplinkIfPolicy2" portMode="np" trunkMode="trunk-on" speed="auto"/>
    <infraFuncP>
      <infraFcAccBndlGrp name="myFcUplinkBundleGroup2">
        <infraRsFcL2IfPol tnFcIfPolName="myFcUplinkIfPolicy2" />
      </infraFcAccBndlGrp>
    </infraFuncP>
    <infraAccPortP name="myFcUplinkPortProfile">
      <infraHPortS name="myFcUplinkSelector" type="range">
        <infraPortBlk name="myUplinkPorts" fromCard="1" toCard="1" fromPort="9" toPort="12"
      />
      <infraRsAccBaseGrp tDn="uni/infra/funcprof/fcaccportgrp-myFcUplinkBundleGroup2" />
    </infraHPortS>
  </infraAccPortP>
  <infraNodeP name="myFcNode1">
    <infraLeafS name="myLeafSelector" type="range">
      <infraNodeBlk name="myLeaf104" from_"104" to_"104" />
    </infraLeafS>
    <infraRsAccPortP tDn="uni/infra/accportprof-myUplinkPorts" />
  </infraNodeP>
</infraInfra>
</polUni>
```

Note When this configuration is applied, a switch reload is required to bring up the ports as FC ports.

Currently only one contiguous range of ports can be converted to FC ports, and this range must be multiple of 4 ending with a port number that is multiple of 4. Examples are 1-4, 1-8, or 21-24.

Step 6 To create the tenant, application profile, EPG and associate the FC bridge domain with the EPG, send a post with XML such as the following example. The example creates a bridge domain myFcBD1 under a target tenant configured to support FC and an application EPG epg1. It associates the EPG with Fibre Channel domain myFcDomain1 and a Fibre Channel path to interface 1/7 on leaf switch 104. Each interface is associated with a VSAN.

Example:

`https://apic-ip-address/api/mo/uni/tn-tenant1.xml`

```
<fvTenant name="tenant1">
  <fvCtx name="myFcVRF"/>
  <fvBD name="myFcBD1" type="fc">
    <fvRsCtx tnFvCtxName="myFcVRF"/>
  </fvBD>
</fvTenant>
```



```

</fvBD>
<fvAp name="app1">
  <fvAEPg name="epg1">
    <fvRsBd tnFvBDName="myFcBD1"/>
    <fvRsDomAtt tDn="uni/fc-myFcDomain1"/>
    <fvRsFcPathAtt tDn="topology/pod-1/paths-104/pathep-[fc1/1]" vsan="vsan-50" vsanMode="native"/>

    <fvRsFcPathAtt tDn="topology/pod-1/paths-104/pathep-[fc1/2]" vsan="vsan-50" vsanMode="native"/>

  </fvAEPg>
</fvAp>
</fvTenant>

```

Step 7 To create a traffic map to pin server ports to uplink ports, send a post with XML such as the following example. The example creates a traffic map to pin server port vFC 1/47 to uplink port FC 1/7:

Example:

<https://apic-ip-address/api/mo/uni/tn-tenant1.xml>

```

<fvTenant name="tenant1">
  <fvAp name="app1">
    <fvAEPg name="epg1">
      <fvRsFcPathAtt tDn="topology/pod-1/paths-104/pathep-[eth1/47]" vsan="vsan-50" vsanMode="native">

        <fcPinningLbl name="label1"/>
      </fvRsFcPathAtt>
    </fvAEPg>
  </fvAp>
</fvTenant>

```

https://apic-ip-address/api/mo/uni/tn-vfc_t1.xml

```

<fvTenant name="tenant1">
  <fcPinningP name="label1">
    <fcRsPinToPath tDn="topology/pod-1/paths-104/pathep-[fc1/7]"/>
  </fcPinningP>
</fvTenant>

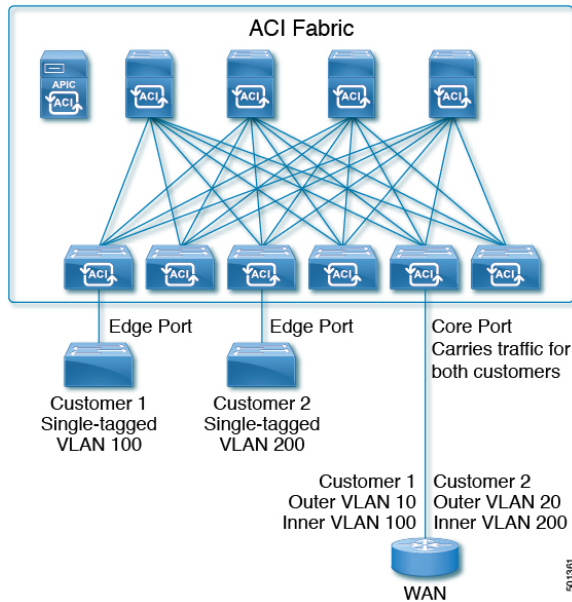
```

Note If traffic map pinning is configured for the first time, the server host port must be shut before configuring the first traffic map.

802.1Q Tunnels

About ACI 802.1Q Tunnels

Figure 23: ACI 802.1Q Tunnels



With Cisco ACI and Cisco APIC Release 2.2(1x) and higher, you can configure 802.1Q tunnels on edge (tunnel) ports to enable point-to-multi-point tunneling of Ethernet frames in the fabric, with Quality of Service (QoS) priority settings. A **Dot1q Tunnel** transports untagged, 802.1Q tagged, and 802.1ad double-tagged frames as-is across the fabric. Each tunnel carries the traffic from a single customer and is associated with a single bridge domain. ACI front panel ports can be part of a **Dot1q Tunnel**. Layer 2 switching is done based on Destination MAC (DMAC) and regular MAC learning is done in the tunnel. Edge-port **Dot1q Tunnels** are supported on second-generation (and later) Cisco Nexus 9000 series switches with "EX" on the end of the switch model name.

With Cisco ACI and Cisco APIC Release 2.3(x) and higher, you can also configure multiple 802.1Q tunnels on the same core port to carry double-tagged traffic from multiple customers, each distinguished with an access encapsulation configured for each 802.1Q tunnel. You can also disable MAC Address Learning on 802.1Q tunnels. Both edge ports and core ports can belong to an 802.1Q tunnel with access encapsulation and disabled MAC Address Learning. Both edge ports and core ports in **Dot1q Tunnels** are supported on third-generation Cisco Nexus 9000 series switches with "FX" and "FX2" on the end of the switch model name.

Terms used in this document may be different in the **Cisco Nexus 9000 Series** documents.

Table 3: 802.1Q Tunnel Terminology

ACI Documents	Cisco Nexus 9000 Series Documents
Edge Port	Tunnel Port
Core Port	Trunk Port

The following guidelines and restrictions apply:

- Layer 2 tunneling of VTP, CDP, LACP, LLDP, and STP protocols is supported with the following restrictions:
 - Link Aggregation Control Protocol (LACP) tunneling functions as expected only with point-to-point tunnels using individual leaf interfaces. It is not supported on port-channels (PCs) or virtual port-channels (vPCs).
 - CDP and LLDP tunneling with PCs or vPCs is not deterministic; it depends on the link it chooses as the traffic destination.
 - To use VTP for Layer 2 protocol tunneling, CDP must be enabled on the tunnel.
 - STP is not supported in an 802.1Q tunnel bridge domain when Layer 2 protocol tunneling is enabled and the bridge domain is deployed on Dot1q Tunnel core ports.
 - ACI leaf switches react to STP TCN packets by flushing the end points in the tunnel bridge domain and flooding them in the bridge domain.
 - CDP and LLDP tunneling with more than two interfaces flood packets on all interfaces.
 - With Cisco APIC Release 2.3(x) or higher, the destination MAC address of Layer 2 protocol packets tunneled from edge to core ports is rewritten as 01-00-0c-cd-cd-d0 and the destination MAC address of Layer 2 protocol packets tunneled from core to edge ports is rewritten with the standard default MAC address for the protocol.
- If a PC or vPC is the only interface in a **Dot1q Tunnel** and it is **deleted** and reconfigured, remove the association of the PC/VPC to the **Dot1q Tunnel** and reconfigure it.
- For 802.1Q tunnels deployed on switches that have EX in the product ID, Ethertype combinations of 0x8100+0x8100, 0x8100+0x88a8, 0x88a8+0x8100, and 0x88a8+0x88a8 for the first two VLAN tags are not supported.

If the tunnels are deployed on a combination of EX and FX or later switches, then this restriction still applies.

If the tunnels are deployed only on switches that have FX or later in the product ID, then this restriction does not apply.
- For core ports, the Ethertypes for double-tagged frames must be 0x8100 followed by 0x8100.
- You can include multiple edge ports and core ports (even across leaf switches) in a **Dot1q Tunnel**.
- An edge port may only be part of one tunnel, but a core port can belong to multiple Dot1q tunnels.
- With Cisco APIC Release 2.3(x) and higher, regular EPGs can be deployed on core ports that are used in 802.1Q tunnels.
- L3Outs are not supported on interfaces enabled for **Dot1q Tunnels**.
- FEX interfaces are not supported as members of a **Dot1q Tunnel**.
- Interfaces configured as breakout ports do not support 802.1Q tunnels.
- Interface-level statistics are supported for interfaces in **Dot1q Tunnels**, but statistics at the tunnel level are not supported.

Configuring 802.1Q Tunnels With Ports Using the REST API

Create a **Dot1q Tunnel**, using ports, and configure the interfaces for it with steps such as the following examples.

Before you begin

Configure the tenant that will use the **Dot1q Tunnel**.

Step 1 Create a **Dot1q Tunnel** using the REST API with XML such as the following example.

The example configures the tunnel with the LLDP Layer 2 tunneling protocol, adds the access encapsulation VLAN, and disables MAC learning in the tunnel.

Example:

```
<fvTnlEPg name="VRF64_dot1q_tunnel" qiqL2ProtTunMask="lldp" accEncap="vlan-10"
  fwdCtrl="mac-learn-disable" >
  <fvRsTnlpathAtt tDn="topology/pod-1/paths-101/pathep-[eth1/13]"/>
</fvTnlEPg>
```

Step 2 Configure a Layer 2 Interface policy with static binding with XML such as the following example.

The example configures a Layer 2 interface policy for edge-switch ports. To configure a policy for core-switch ports, use `corePort` instead of `edgePort` in the `l2IfPol` MO.

Example:

```
<l2IfPol name="VRF64_L2_int_pol" qinq="edgePort" />
```

Step 3 Apply the Layer 2 Interface policy to a Leaf Access Port Policy Group with XML such as the following example.

Example:

```
<infraAccPortGrp name="VRF64_L2_Port_Pol_Group" >
  <infraRsL2IfPol tnL2IfPolName="VRF64_L2_int_pol"/>
</infraAccPortGrp>
```

Step 4 Configure a Leaf Profile with an Interface Selector with XML such as the following example:

Example:

```
<infraAccPortP name="VRF64_dot1q_leaf_profile" >
  <infraHPortS name="vrf64_access_port_selector" type="range">
    <infraPortBlk name="block2" toPort="15" toCard="1" fromPort="13" fromCard="1"/>
    <infraRsAccBaseGrp tDn="uni/infra/funcprof/accportgrp-VRF64_L2_Port_Pol_Group" />
  </infraHPortS>
</infraAccPortP>
```

Example

The following example shows the port configuration for edge ports in two posts.

XML with Post 1:

```
<polUni>
  <infraInfra>
    <l2IfPol name="testL2IfPol" qinq="edgePort"/>
    <infraNodeP name="Node_101_phys">
```

```

    <infraLeafS name="phys101" type="range">
      <infraNodeBlk name="test" from_"=101" to_"=101"/>
    </infraLeafS>
    <infraRsAccPortP tDn="uni/infra/accportprof-phys21"/>
  </infraNodeP>
  <infraAccPortP name="phys21">
    <infraHPortS name="physHPortS" type="range">
      <infraPortBlk name="phys21" fromCard="1" toCard="1" fromPort="21" toPort="21"/>
      <infraRsAccBaseGrp tDn="uni/infra/funcprof/accportgrp-21"/>
    </infraHPortS>
  </infraAccPortP>
  <infraFuncP>
    <infraAccPortGrp name="21">
      <infraRsL2IfPol tnL2IfPolName="testL2IfPol"/>
      <infraRsAttEntP tDn="uni/infra/attentp-AttEntityProf1701"/>
    </infraAccPortGrp>
  </infraFuncP>
  <l2IfPol name='testL2IfPol' qinq='edgePort' />
  <infraAttEntityP name="AttEntityProf1701">
    <infraRsDomP tDn="uni/phys-dom1701"/>
  </infraAttEntityP>
</infraInfra>
</polUni>

```

XML with Post 2:

```

<polUni>
  <fvTenant dn="uni/tn-Coke" name="Coke">
    <fvTnlEPg name="WEB5" qiqL2ProtTunMask="lldp" accEncap="vlan-10"
    fwdCtrl="mac-learn-disable" >
      <fvRsTnlpathAtt tDn="topology/pod-1/paths-101/pathep-[eth1/21]"/>
    </fvTnlEPg>
  </fvTenant>
</polUni>

```

Configuring 802.1Q Tunnels With PCs Using the REST API

Create a **Dot1q Tunnel**, using PCs, and configure the interfaces for it with steps such as the following examples.

Before you begin

Configure the tenant that will use the **Dot1q Tunnel**.

Step 1 Create a **Dot1q Tunnel** using the REST API with XML such as the following example.

The example configures the tunnel with the LLDP Layer 2 tunneling protocol, adds the access encapsulation VLAN, and disables MAC learning in the tunnel.

Example:

```

<fvTnlEPg name="WEB" qiqL2ProtTunMask=lldp accEncap="vlan-10" fwdCtrl="mac-learn-disable" >
  <fvRsTnlpathAtt tDn="topology/pod-1/paths-101/pathep-[po2]"/>
</fvTnlEPg>

```

Step 2 Configure a Layer 2 Interface policy with static binding with XML such as the following example.

The example configures a Layer 2 interface policy for edge-switch ports. To configure a Layer 2 interface policy for core-switch ports, use `corePort` instead of `edgePort` in the `l2IfPol` MO.

Example:

```
<l2IfPol name="testL2IfPol" qinq="edgePort"/>
```

Step 3 Apply the Layer 2 Interface policy to a PC Interface Policy Group with XML such as the following:

Example:

```
<infraAccBndlGrp name="po2" lagT="link">
  <infraRsL2IfPol tnL2IfPolName="testL2IfPol"/>
</infraAccBndlGrp>
```

Step 4 Configure a Leaf Profile with an Interface Selector with XML such as the following:

Example:

```
<infraAccPortP name="PC">
  <infraHPortS name="allow" type="range">
    <infraPortBlk name="block2" fromCard="1" toCard="1" fromPort="10" toPort="11" />
    <infraRsAccBaseGrp tDn="uni/infra/funcprof/accbundle-po2"/>
  </infraHPortS>
</infraAccPortP>
```

Example

The following example shows the PC configuration in two posts.

This example configures the PC ports as edge ports. To configure them as core ports, use `corePort` instead of `edgePort` in the `l2IfPol` MO, in Post 1.

XML with Post 1:

```
<infraInfra dn="uni/infra">
  <infraNodeP name="bLeaf3">
    <infraLeafS name="leafs3" type="range">
      <infraNodeBlk name="nblk3" from_"101" to_"101">
        </infraNodeBlk>
      </infraLeafS>
      <infraRsAccPortP tDn="uni/infra/accportprof-shipping3"/>
    </infraNodeP>
    <infraAccPortP name="shipping3">
      <infraHPortS name="pselc3" type="range">
        <infraPortBlk name="blk3" fromCard="1" toCard="1" fromPort="24" toPort="25"/>
        <infraRsAccBaseGrp tDn="uni/infra/funcprof/accbundle-accountingLag3" />
      </infraHPortS>
    </infraAccPortP>
  </infraInfra>
  <infraFuncP>
    <infraAccBndlGrp name="accountingLag3" lagT='link'>
      <infraRsAttEntP tDn="uni/infra/attentp-default"/>
      <infraRsLacpPol tnLacpLagPolName='accountingLacp3' />
      <infraRsL2IfPol tnL2IfPolName="testL2IfPol3"/>
    </infraAccBndlGrp>
  </infraFuncP>
  <lacpLagPol name='accountingLacp3' ctrl='15' descr='accounting' maxLinks='14' minLinks='1'
  mode='active' />
  <l2IfPol name='testL2IfPol3' qinq='edgePort' />
  <infraAttEntityP name="default">
    </infraAttEntityP>
  </infraInfra>
```

XML with Post 2:

```

<polUni>
  <fvTenant dn="uni/tn-Coke" name="Coke">
    <!-- bridge domain -->
    <fvTnlEPg name="WEB6" qiqL2ProtTunMask="lldp" accEncap="vlan-10"
    fwdCtrl="mac-learn-disable" >
      <fvRsTnlpathAtt tDn="topology/pod-1/paths-101/pathep-[accountingLag1]"/>
    </fvTnlEPg>
  </fvTenant>
</polUni>

```

Configuring 802.1 Q Tunnels With vPCs Using the REST API

Create a **Dot1q Tunnel**, using vPCs, and configure the interfaces for it with steps such as the following examples.

Before you begin

Configure the tenant that will use the **Dot1q Tunnel**.

Step 1 Create an 802.1Q tunnel using the REST API with XML such as the following example.

The example configures the tunnel with a Layer 2 tunneling protocol, adds the access encapsulation VLAN, and disables MAC learning in the tunnel.

Example:

```

<fvTnlEPg name="WEB" qiqL2ProtTunMask=lldp accEncap="vlan-10" fwdCtrl="mac-learn-disable" >
  <fvRsTnlpathAtt tDn="topology/pod-1/protpaths-101-102/pathep-[po4]" />
</fvTnlEPg>

```

Step 2 Configure a Layer 2 interface policy with static binding with XML such as the following example.

The example configures a Layer 2 interface policy for edge-switch ports. To configure a Layer 2 interface policy for core-switch ports, use the `qinq="corePort"` port type.

Example:

```

<l2IfPol name="testL2IfPol" qinq="edgePort"/>

```

Step 3 Apply the Layer 2 Interface policy to a VPC Interface Policy Group with XML such as the following:

Example:

```

<infraAccBndlGrp name="po4" lagT="node">
  <infraRsL2IfPol tnL2IfPolName="testL2IfPol"/>
</infraAccBndlGrp>

```

Step 4 Configure a Leaf Profile with an Interface Selector with XML such as the following:

Example:

```

<infraAccPortP name="VPC">
  <infraHPortS name="allow" type="range">
    <infraPortBlk name="block2" fromCard="1" toCard="1" fromPort="10" toPort="11" />
    <infraRsAccBaseGrp tDn="uni/infra/funcprof/accbundle-po4"/>
  </infraHPortS>
</infraAccPortP>

```

Example

The following example shows the vPC configuration in three posts.

This example configures the vPC ports as edge ports. To configure them as core ports, use `corePort` instead of `edgePort` in the `l2IfPol` MO, in Post 2

XML with Post 1:

```
<polUni>
  <fabricInst>
    <fabricProtPol pairT="explicit">
      <fabricExplicitGEp name="101-102-vpc1" id="30">
        <fabricNodePEp id="101"/>
        <fabricNodePEp id="102"/>
      </fabricExplicitGEp>
    </fabricProtPol>
  </fabricInst>
</polUni>
```

XML with Post 2:

```
<infraInfra dn="uni/infra">
  <infraNodeP name="bLeaf1">
    <infraLeafS name="leafs" type="range">
      <infraNodeBlk name="nblk" from_"101" to_"101">
      </infraNodeBlk>
    </infraLeafS>
    <infraRsAccPortP tDn="uni/infra/accportprof-shipping1"/>
  </infraNodeP>

  <infraNodeP name="bLeaf2">
    <infraLeafS name="leafs" type="range">
      <infraNodeBlk name="nblk" from_"102" to_"102">
      </infraNodeBlk>
    </infraLeafS>
    <infraRsAccPortP tDn="uni/infra/accportprof-shipping2"/>
  </infraNodeP>

  <infraAccPortP name="shipping1">
    <infraHPortS name="pselc" type="range">
      <infraPortBlk name="blk" fromCard="1" toCard="1" fromPort="4" toPort="4"/>
      <infraRsAccBaseGrp tDn="uni/infra/funcprof/accbundle-accountingLag1" />
    </infraHPortS>
  </infraAccPortP>

  <infraAccPortP name="shipping2">
    <infraHPortS name="pselc" type="range">
      <infraPortBlk name="blk" fromCard="1" toCard="1" fromPort="2" toPort="2"/>
      <infraRsAccBaseGrp tDn="uni/infra/funcprof/accbundle-accountingLag2" />
    </infraHPortS>
  </infraAccPortP>

  <infraFuncP>
    <infraAccBndlGrp name="accountingLag1" lagT='node'>
      <infraRsAttEntP tDn="uni/infra/attentp-default"/>
      <infraRsLacpPol tnLacpLagPolName='accountingLacp1'/>
      <infraRsL2IfPol tnL2IfPolName="testL2IfPol"/>
    </infraAccBndlGrp>
    <infraAccBndlGrp name="accountingLag2" lagT='node'>
      <infraRsAttEntP tDn="uni/infra/attentp-default"/>
      <infraRsLacpPol tnLacpLagPolName='accountingLacp1'/>
      <infraRsL2IfPol tnL2IfPolName="testL2IfPol"/>
    </infraAccBndlGrp>
  </infraFuncP>
```



```

    </infraAccBndlGrp>
</infraFuncP>
<lacpLagPol name='accountingLacpl' ctrl='15' descr='accounting' maxLinks='14' minLinks='1'
mode='active' />
<l2IfPol name='testL2IfPol' qinq='edgePort'/>

    <infraAttEntityP name="default">
</infraAttEntityP>
</infraInfra>

```

XML with Post 3:

```

<polUni>
  <fvTenant dn="uni/tn-Coke" name="Coke">
    <!-- bridge domain -->
    <fvTnlEPg name="WEB6" qiqL2ProtTunMask="lldp" accEncap="vlan-10"
fwdCtrl="mac-learn-disable" >
      <fvRsTnlpathAtt tDn="topology/pod-1/protopaths-101-102/pathep-[accountingLag2]"/>
    </fvTnlEPg>
  </fvTenant>
</polUni>

```

Breakout Ports

Configuration of Dynamic Breakout Ports

Breakout cables are suitable for very short links and offer a cost effective way to connect within racks and across adjacent racks.

Breakout enables a 40 Gigabit (Gb) port to be split into four independent and logical 10Gb ports or a 100Gb port to be split into four independent and logical 25Gb ports.

Before you configure breakout ports, connect a 40Gb port to four 10Gb ports or a 100Gb port to four 25Gb ports with one of the following cables:

- Cisco QSFP-4SFP10G
- Cisco QSFP-4SFP25G
- Cisco QSFP-4X10G-AOC
- MPO to breakout splitter cable with QSFP-40G-SR4 and 4 x SFP-10G-SR on the ends
- MPO to breakout splitter cable with QSFP-100G-SR4-S and 4 x SFP-25G-SR-S on the ends



Note For the supported optics and cables, see the *Cisco Optics-to-Device Compatibility Matrix*:
<https://tmgmatrix.cisco.com/>

The 40Gb to 10Gb dynamic breakout feature is supported on the access facing ports of the following switches:

- N9K-C9332PQ
- N9K-C93180LC-EX

- N9K-C9336C-FX2
- N9K-C93360YC-FX2
- N9K-C93216TC-FX2

The 100Gb to 25Gb breakout feature is supported on the access facing ports of the following switches:

- N9K-C93180LC-EX
- N9K-C9336C-FX2
- N9K-C93180YC-FX
- N9K-C93360YC-FX2
- N9K-C93216TC-FX2

Observe the following guidelines and limitations:

- For the Cisco N9K-C9332PQ switch, you can configure ports 1 to 26 as downlink ports. Of those ports, breakout ports can be configured on port 1 to 12 and 15 to 26. Ports 13 and 14 do not support breakout.
- Breakout ports are supported only on downlinks and converted downlinks.
 - Starting in Cisco Application Policy Infrastructure Controller (APIC) release 3.2(1), dynamic breakouts (both 100Gb and 40Gb) are supported on profiled QSFP ports on the Cisco N9K-C93180YC-FX switch.
 - Starting in Cisco APIC release 4.1(2), dynamic breakouts (both 100Gb and 40Gb) are supported on profiled QSFP ports on the Cisco N9K-C93216TC-FX2 switch.
 - Starting in Cisco APIC release 4.1(2), dynamic breakouts (both 100Gb and 40Gb) are supported on profiled QSFP ports on the Cisco N9K-C93360YC-FX2 switch.
- Breakout ports cannot be used for Cisco APIC connectivity.
- Fast Link Failover policies are not supported on the same port with the dynamic breakout feature.
- Breakout subports can be used in the same way other port types in the policy model are used.
- When a port is enabled for dynamic breakout, other policies (except monitoring policies) on the parent port are no longer valid.
- When a port is enabled for dynamic breakout, other EPG deployments on the parent port are no longer valid.
- A breakout sub-port can not be further broken out using a breakout policy group.
- If the LACP transmit rate on port channels that have breakout sub-ports need to be changed, then all the port channels that include breakout sub-ports need to use the same LACP transmit rate configuration. You can configure an override policy to set the transmit rate as follows:
 1. Configure/change the default port channel member policy to include Fast Transmit Rate (**Fabric > Access Policies > Policies > Interface > Port Channel Member**).
 2. Configure all the PC/vPC interface policy groups to include the above default port channel member policy under the override policy groups (**Fabric > Access Policies > Interfaces > Leaf Interfaces > Policy Groups > PC/vPC Interface**).

Configuring Dynamic Breakout Ports Using the REST API

Configure a Breakout Leaf Port with an Leaf Interface Profile, associate the profile with a switch, and configure the sub ports with the following steps.

For switch support for the breakout feature, see [Configuration of Dynamic Breakout Ports, on page 271](#).

Procedure

Before you begin

- The ACI fabric is installed, APIC controllers are online, and the APIC cluster is formed and healthy.
- An APIC fabric administrator account is available that will enable creating the necessary fabric infrastructure configurations.
- The target leaf switches are registered in the ACI fabric and available.
- The 40GE or 100GE leaf switch ports are connected with Cisco breakout cables to the downlink ports.

Step 1 Configure a breakout policy group for the breakout port with JSON, such as the following example:

Example:

In this example, we create an interface profile 'brkout44' with the only port 44 underneath its port selector. The port selector is pointed to a breakout policy group 'new-brkoutPol'.

```
{
  "infraAccPortP": {
    "attributes": {
      "dn": "uni/infra/accportprof-brkout44",
      "name": "brkout44",
      "rn": "accportprof-brkout44",
      "status": "created,modified"
    },
    "children": [ {
      "infraHPortS": {
        "attributes": {
          "dn": "uni/infra/accportprof-brkout44/hports-new-brkoutPol-typ-range",
          "name": "new-brkoutPol",
          "rn": "hports-new-brkoutPol-typ-range",
          "status": "created,modified"
        },
        "children": [ {
          "infraPortBlk": {
            "attributes": {
              "dn": "uni/infra/accportprof-brkout44/hports-new-brkoutPol-typ-range/portblk-block2",
              "fromPort": "44",
              "toPort": "44",
              "name": "block2",
              "rn": "portblk-block2",
              "status": "created,modified"
            },
            "children": [ ] }
          }, {
            "infraRsAccBaseGrp": {
              "attributes": {
                "tDn": "uni/infra/funcprof/brkoutportgrp-new-brkoutPol",
                "status": "created,modified"
              }
            }
          }
        ]
      }
    }
  }
}
```


This example configures subports 1/44/1, 1/44/2, 1/44/3, 1/44/4 on switch 1017, for instance, in the example below, we configure interface 1/44/3. It also creates the `infraSubPortBlk` object instead of the `infraPortBlk` object.

```
{
  "infraAccPortP": {
    "attributes": {
      "dn": "uni/infra/accportprof-brkout44",
      "name": "brkouttest1",
      "rn": "accportprof-brkout44",
      "status": "created,modified"
    },
    "children": [{
      "infraHPorts": {
        "attributes": {
          "dn": "uni/infra/accportprof-brkout44/hports-sell-typ-range",
          "name": "sell",
          "rn": "hports-sell-typ-range",
          "status": "created,modified"
        },
        "children": [{
          "infraSubPortBlk": {
            "attributes": {
              "dn": "uni/infra/accportprof-brkout44/hports-sell-typ-range/subportblk-block2",
              "fromPort": "44",
              "toPort": "44",
              "fromSubPort": "3",
              "toSubPort": "3",
              "name": "block2",
              "rn": "subportblk-block2",
              "status": "created"
            },
            "children": []
          },
          {
            "infraRsAccBaseGrp": {
              "attributes": {
                "tDn": "uni/infra/funcprof/accportgrp-p1",
                "status": "created,modified"
              },
              "children": []
            }
          }
        ]
      }
    ]
  }
}
```

Step 4 Deploy an EPG on a specific port.

Example:

```
<fvTenant name="<tenant_name>" dn="uni/tn-test1" >
  <fvCtx name="<network_name>" pcEnfPref="enforced" knwMcastAct="permit"/>
  <fvBD name="<bridge_domain_name>" unkMcastAct="flood" >
    <fvRsCtx tnFvCtxName="<network_name>"/>
  </fvBD>
  <fvAp name="<application_profile>" >
    <fvAEPg name="<epg_name>" >
      <fvRsPathAtt tDn="topology/pod-1/paths-1017/pathep-[eth1/13]" mode="regular"
instrImedcy="immediate" encap="vlan-20"/>
    </fvAEPg>
  </fvAp>
</fvTenant>
```

```
</fvAp>  
</fvTenant>
```

Port Profiles to Change Uplinks to Downlinks and Downlinks to Uplinks

Configuring Port Profiles

Prior to Cisco Application Policy Infrastructure Controller (APIC) release 3.1(1), conversion from uplink port to downlink port or downlink port to uplink port (in a port profile) was not supported on Cisco ACI leaf switches. Starting with Cisco APIC release 3.1(1), uplink and downlink conversion is supported on Cisco Nexus 9000 series switches with names that end in EX or FX, and later (for example, N9K-C9348GC-FXP or N9K-C93240YC-FX2). A FEX connected to converted downlinks is also supported.

This functionality is supported on the following Cisco switches:

- N9K-C9348GC-FXP (does not support FEX)
- N9K-C93180LC-EX
- N9K-C93180YC-FX and N9K-93180YC-EX
- N9K-C93108TC-EX and N9K-C93108TC-FX (only uplink to downlink conversion is supported)
- N9K-C9336C-FX2 (only downlink to uplink conversion is supported)
- N9K-C93240YC-FX2

When an uplink port is converted to a downlink port, it acquires the same capabilities as any other downlink port.

Restrictions

- Fast Link Failover policies and port profiles are not supported on the same port. If port profile is enabled, Fast Link Failover cannot be enabled or vice versa.
- The last 2 uplink ports of supported leaf switches cannot be converted to downlink ports (they are reserved for uplink connections.)
- Dynamic breakouts (both 100Gb and 40Gb) are supported on profiled QSFP ports on the N9K-C93180YC-FX switch. Breakout and port profile are supported together for conversion of uplink to downlink on ports 49-52. Breakout (both **10g-4x** or **25g-4x** options) is supported on downlink profiled ports.
- The N9K-C9348GC-FXP does not support FEX.
- Breakout is supported only on downlink ports, and not on fabric ports that are connected to other switches.
- A Cisco ACI leaf switch cannot have more than 56 fabric links.

- Reloading a switch after changing a switch's port profile configuration interrupts traffic through the data plane.

Guidelines

In converting uplinks to downlinks and downlinks to uplinks, consider the following guidelines.

Subject	Guideline
Decommissioning nodes with port profiles	<p>If a decommissioned node has the port profile feature deployed on it, the port conversions are not removed even after decommissioning the node. It is necessary to manually delete the configurations after decommission, for the ports to return to the default state. To do this, log onto the switch, run the <code>setup-clean-config.sh</code> script, and wait for it to run. Then, enter the <code>reload</code> command. Optionally, you can specify <code>-k</code> with the <code>setup-clean-config.sh</code> script to allow the port-profile setting to persist across the reload, making an additional reboot unnecessary.</p>
FIPS	<p>When you enable or disable Federal Information Processing Standards (FIPS) on a Cisco ACI fabric, you must reload each of the switches in the fabric for the change to take effect. The configured scale profile setting is lost when you issue the first reload after changing the FIPS configuration. The switch remains operational, but it uses the default scale profile. This issue does not happen on subsequent reloads if the FIPS configuration has not changed.</p> <p>FIPS is supported on Cisco NX-OS release 13.1(1) or later.</p> <p>If you must downgrade the firmware from a release that supports FIPS to a release that does not support FIPS, you must first disable FIPS on the Cisco ACI fabric and reload all the switches in the fabric for the FIPS configuration change.</p>

Subject	Guideline
Maximum uplink port limit	<p>When the maximum uplink port limit is reached and ports 25 and 27 are converted from uplink to downlink and back to uplink on Cisco 93180LC-EX switches:</p> <p>On Cisco 93180LC-EX Switches, ports 25 and 27 are the native uplink ports. Using the port profile, if you convert port 25 and 27 to downlink ports, ports 29, 30, 31, and 32 are still available as four native uplink ports. Because of the threshold on the number of ports (which is maximum of 12 ports) that can be converted, you can convert 8 more downlink ports to uplink ports. For example, ports 1, 3, 5, 7, 9, 13, 15, 17 are converted to uplink ports and ports 29, 30, 31 and 32 are the 4 native uplink ports (the maximum uplink port limit on Cisco 93180LC-EX switches).</p> <p>When the switch is in this state and if the port profile configuration is deleted on ports 25 and 27, ports 25 and 27 are converted back to uplink ports, but there are already 12 uplink ports on the switch (as mentioned earlier). To accommodate ports 25 and 27 as uplink ports, 2 random ports from the port range 1, 3, 5, 7, 9, 13, 15, 17 are denied the uplink conversion and this situation cannot be controlled by the user.</p> <p>Therefore, it is mandatory to clear all the faults before reloading the leaf node to avoid any unexpected behavior regarding the port type. It should be noted that if a node is reloaded without clearing the port profile faults, especially when there is a fault related to limit-exceed, the port might not be in an expected operational state.</p>

Breakout Limitations

Switch	Releases	Limitations
N9K-C9332PQ	Cisco APIC 2.2 (1n) and higher	<ul style="list-style-type: none"> • 40Gb dynamic breakouts into 4X10Gb ports are supported. • Ports 13 and 14 do not support breakouts. • Port profiles and breakouts are not supported on the same port.
N9K-C93180LC-EX	Cisco APIC 3.1(1i) and higher	<ul style="list-style-type: none"> • 40Gb and 100Gb dynamic breakouts are supported on ports 1 through 24 on odd numbered ports. • When the top ports (odd ports) are broken out, then the bottom ports (even ports) are error disabled. • Port profiles and breakouts are not supported on the same port.

Switch	Releases	Limitations
N9K-C9336C-FX2	Cisco APIC 3.2(11) and higher	<ul style="list-style-type: none"> • 40Gb and 100Gb dynamic breakouts are supported on ports 1 through 30. • Port profiles and breakouts are not supported on the same port.
N9K-C93180YC-FX	Cisco APIC 3.2(11) and higher	<ul style="list-style-type: none"> • 40Gb and 100Gb dynamic breakouts are supported on ports 49 through 52, when they are on profiled QSFP ports. To use them for dynamic breakout, perform the following steps: <ul style="list-style-type: none"> • Convert ports 49-52 to front panel ports (downlinks). • Perform a port-profile reload, using one of the following methods: <ul style="list-style-type: none"> • In the Cisco APIC GUI, navigate to Fabric > Inventory > Pod > Leaf, right-click Chassis and choose Reload. • In the iBash CLI, enter the reload command. • Apply breakouts on the profiled ports 49-52. • Ports 53 and 54 do not support either port profiles or breakouts.
N9K-C93240YC-FX2	Cisco APIC 4.0(1) and higher	Breakout is not supported on converted downlinks.

Port Profile Configuration Summary

The following table summarizes supported uplinks and downlinks for the switches that support port profile conversions from Uplink to Downlink and Downlink to Uplink.

Switch Model	Default Links	Max Uplinks (Fabric Ports)	Max Downlinks (Server Ports)	Release Supported
N9K-C9348GC-FXP	48 x 100M/1G BASE-T downlinks 4 x 10/25-Gbps SFP28 downlinks 2 x 40/100-Gbps QSFP28 uplinks	48 x 100M/1G BASE-T downlinks 4 x 10/25-Gbps SFP28 uplinks 2 x 40/100-Gbps QSFP28 uplinks	Same as default	3.1(1i)

Switch Model	Default Links	Max Uplinks (Fabric Ports)	Max Downlinks (Server Ports)	Release Supported
N9K-C93180LC-EX	24 x 40-Gbps QSFP 28 downlinks 6 x 40/100-Gbps QSFP 28 uplinks Or 12 x 100-Gbps QSFP 28 downlinks 6 x 40/100-Gbps QSFP 28 uplinks	12 x 40-Gbps QSFP 28 downlinks 12 x 40/100-Gbps QSFP 28 uplinks Or 6 x 100-Gbps QSFP 28 downlinks 12 x 40/100-Gbps QSFP 28 uplinks	4 x 40-Gbps QSFP 28 downlinks 2 x 40/100-Gbps QSFP 28 downlinks 4 x 40/100-Gbps uplinks Or 12 x 100-Gbps QSFP 28 downlinks 2 x 40/100-Gbps QSFP 28 downlinks 4 x 40/100-Gbps uplinks	3.1(1i)
N9K-C93180YC-EX N9K-C93180YC-FX	48 x 10/25-Gbps fiber downlinks 6 x 40/100-Gbps QSFP28 uplinks	48 x 10/25-Gbps fiber downlinks 6 x 40/100-Gbps QSFP28 uplinks	48 x 10/25-Gbps fiber downlinks 4 x 40/100-Gbps QSFP28 downlinks 2 x 40/100-Gbps QSFP28 uplinks	3.1(1i)
N9K-C93108TC-EX N9K-C93108TC-FX	48 x 10GBASE-T downlinks 6 x 40/100-Gbps QSFP28 uplinks	Same as default	48 x 10/25-Gbps fiber downlinks 4 x 40/100-Gbps QSFP28 downlinks 2 x 40/100-Gbps QSFP28 uplinks	3.1(1i)
N9K-C9336C-FX2	30 x 40/100-Gbps QSFP28 downlinks 6 x 40/100-Gbps QSFP28 uplinks	18 x 40/100-Gbps QSFP28 downlinks 18 x 40/100-Gbps QSFP28 uplinks	Same as default	3.2(11)

Configuring a Port Profile Using the REST API

Before you begin

- The ACI fabric is installed, APIC controllers are online, and the APIC cluster is formed and healthy.
- An APIC fabric administrator account is available that will enable creating or modifying the necessary fabric infrastructure configurations.
- The target leaf switches are registered in the ACI fabric and available.

Step 1 To create a port profile that converts a downlink to an uplink, send a post with XML such as the following:

```
<!-- /api/node/mo/uni/infra/prtdirec.xml -->  
<infraRsPortDirection tDn="topology/pod-1/paths-106/pathep-[eth1/7]" direc="UpLink" />
```

Step 2 To create a port profile that converts an uplink to a downlink, send a post with XML such as the following:

Example:

```
<!-- /api/node/mo/uni/infra/prtdirec.xml -->  
<infraRsPortDirection tDn="topology/pod-1/paths-106/pathep-[eth1/52]" direc="DownLink" />
```

IGMP Snooping

About Cisco APIC and IGMP Snooping

IGMP snooping is the process of listening to Internet Group Management Protocol (IGMP) network traffic. The feature allows a network switch to listen in on the IGMP conversation between hosts and routers and filter multicasts links that do not need them, thus controlling which ports receive specific multicast traffic.

Cisco APIC provides support for the full IGMP snooping feature included on a traditional switch such as the N9000 standalone.

- Policy-based IGMP snooping configuration per bridge domain

APIC enables you to configure a policy in which you enable, disable, or customize the properties of IGMP Snooping on a per bridge-domain basis. You can then apply that policy to one or multiple bridge domains.

- Static port group implementation

IGMP static port grouping enables you to pre-provision ports, already statically-assigned to an application EPG, as the switch ports to receive and process IGMP multicast traffic. This pre-provisioning prevents the join latency which normally occurs when the IGMP snooping stack learns ports dynamically.

Static group membership can be pre-provisioned only on static ports (also called, *static-binding ports*) assigned to an application EPG.

- Access group configuration for application EPGs

An “access-group” is used to control what streams can be joined behind a given port.

An access-group configuration can be applied on interfaces that are statically assigned to an application EPG in order to ensure that the configuration can be applied on ports that will actually belong to the that EPG.

Only Route-map-based access groups are allowed.



Note You can use **vzAny** to enable protocols such as IGMP Snooping for all the EPGs in a VRF. For more information about **vzAny**, see [Use vzAny to Automatically Apply Communication Rules to all EPGs in a VRF](#).

To use **vzAny**, navigate to **Tenants** > *tenant-name* > **Networking** > **VRFs** > *vrf-name* > **EPG Collection for VRF**.

How IGMP Snooping is Implemented in the ACI Fabric

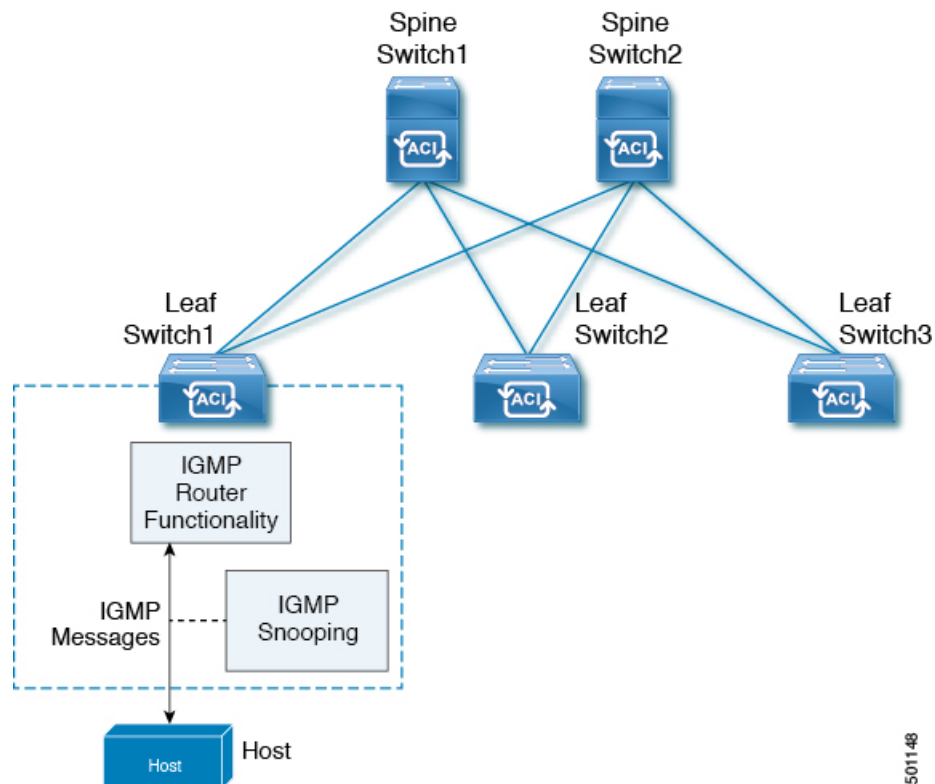


Note We recommend that you do not disable IGMP snooping on bridge domains. If you disable IGMP snooping, you may see reduced multicast performance because of excessive false flooding within the bridge domain.

IGMP snooping software examines IP multicast traffic within a bridge domain to discover the ports where interested receivers reside. Using the port information, IGMP snooping can reduce bandwidth consumption in a multi-access bridge domain environment to avoid flooding the entire bridge domain. By default, IGMP snooping is enabled on the bridge domain.

This figure shows the IGMP routing functions and IGMP snooping functions both contained on an ACI leaf switch with connectivity to a host. The IGMP snooping feature snoops the IGMP membership reports, and leaves messages and forwards them only when necessary to the IGMP router function.

Figure 24: IGMP Snooping function



501148

IGMP snooping operates upon IGMPv1, IGMPv2, and IGMPv3 control plane packets where Layer 3 control plane packets are intercepted and influence the Layer 2 forwarding behavior.

IGMP snooping has the following proprietary features:

- Source filtering that allows forwarding of multicast packets based on destination and source IP addresses
- Multicast forwarding based on IP addresses rather than the MAC address
- Multicast forwarding alternately based on the MAC address

The ACI fabric supports IGMP snooping only in proxy-reporting mode, in accordance with the guidelines provided in Section 2.1.1, "IGMP Forwarding Rules," in RFC 4541:

IGMP networks may also include devices that implement "proxy-reporting", in which reports received from downstream hosts are summarized and used to build internal membership states. Such proxy-reporting devices may use the all-zeros IP Source-Address when forwarding any summarized reports upstream. For this reason, IGMP membership reports received by the snooping switch must not be rejected because the source IP address is set to 0.0.0.0.

As a result, the ACI fabric will send IGMP reports with the source IP address of 0.0.0.0.



Note For more information about IGMP snooping, see RFC 4541.

Virtualization Support

You can define multiple virtual routing and forwarding (VRF) instances for IGMP snooping.

On leaf switches, you can use the **show** commands with a VRF argument to provide a context for the information displayed. The default VRF is used if no VRF argument is supplied.

Configuring and Assigning an IGMP Snooping Policy to a Bridge Domain using the REST API

SUMMARY STEPS

1. To configure an IGMP Snooping policy and assign it to a bridge domain, send a post with XML such as the following example:

DETAILED STEPS

To configure an IGMP Snooping policy and assign it to a bridge domain, send a post with XML such as the following example:

Example:

```

https://apic-ip-address/api/node/mo/uni/.xml
<fvTenant name="mcast_tenant1">

<!-- Create an IGMP snooping template, and provide the options -->
<igmpSnoopPol name="igmp_snp_bd_21"
  adminSt="enabled"
  lastMbrIntvl="1"
  queryIntvl="125"
  rspIntvl="10"
  startQueryCnt="2"
  startQueryIntvl="31"
  />
<fvCtx name="ip_video"/>

<fvBD name="bd_21">
  <fvRsCtx tnFvCtxName="ip_video"/>

  <!-- Bind IGMP snooping to a BD -->
  <fvRsIgmpsn tnIgmpSnoopPolName="igmp_snp_bd_21"/>
</fvBD></fvTenant>

```

This example creates and configures the IGMP Snooping policy, `igmp_snp_bd_12` with the following properties, and binds the IGMP policy, `igmp_snp_bd_21`, to bridge domain, `bd_21`:

- Administrative state is enabled
- Last Member Query Interval is the default 1 second.
- Query Interval is the default 125.
- Query Response interval is the default 10 seconds
- The Start Query Count is the default 2 messages
- The Start Query interval is 35 seconds.

Enabling Group Access to IGMP Snooping and Multicast using the REST API

After you have enabled IGMP snooping and multicast on ports that have been statically assigned to an EPG, you can then create and assign access groups of users that are permitted or denied access to the IGMP snooping and multicast traffic enabled on those ports.

To define the access group, `F23broker`, send a post with XML such as in the following example.

The example configures access group `F23broker`, associated with tenant `_A`, `Rmap_A`, application `_A`, `epg_A`, on leaf 102, interface 1/10, VLAN 202. By association with `Rmap_A`, the access group `F23broker` has access to multicast traffic received at multicast address 226.1.1.1/24 and is denied access to traffic received at multicast address 227.1.1.1/24.

Example:

```

<!-- api/node/mo/uni/.xml --> <fvTenant name="tenant_A"> <pimRouteMapPol name="Rmap_A"> <pimRouteMapEntry
action="permit" grp="226.1.1.1/24" order="10"/> <pimRouteMapEntry action="deny" grp="227.1.1.1/24" order="20"/>
</pimRouteMapPol> <fvAp name="application_A"> <fvAEPg name="epg_A"> <fvRsPathAtt encap="vlan-202"
instrImedcy="immediate" mode="regular" tDn="topology/pod-1/paths-102/pathep-[eth1/10]"> <!-- IGMP snooping

```

```
access group case --> <igmpSnoopAccessGroup name="F23broker"> <igmpRsSnoopAccessGroupFilterRMap
tnPimRouteMapPolName="Rmap_A"/> </igmpSnoopAccessGroup> </fvRsPathAtt> </fvAEPg> </fvAp> </fvTenant>
```

Enabling IGMP Snooping and Multicast on Static Ports Using the REST API

You can enable IGMP snooping and multicast processing on ports that have been statically assigned to an EPG. You can create and assign access groups of users that are permitted or denied access to the IGMP snoop and multicast traffic enabled on those ports.

SUMMARY STEPS

1. To configure application EPGs with static ports, enable those ports to receive and process IGMP snooping and multicast traffic, and assign groups to access or be denied access to that traffic, send a post with XML such as the following example.

DETAILED STEPS

To configure application EPGs with static ports, enable those ports to receive and process IGMP snooping and multicast traffic, and assign groups to access or be denied access to that traffic, send a post with XML such as the following example.

In the following example, IGMP snooping is enabled on `leaf 102` interface `1/10` on VLAN 202. Multicast IP addresses `224.1.1.1` and `225.1.1.1` are associated with this port.

Example:

```
https://apic-ip-address/api/node/mo/uni/.xml
<fvTenant name="tenant_A">
  <fvAp name="application">
    <fvAEPg name="epg_A">
      <fvRsPathAtt encap="vlan-202" instrImedcy="immediate" mode="regular"
tDn="topology/pod-1/paths-102/pathep-[eth1/10]">
        <!-- IGMP snooping static group case -->
        <igmpSnoopStaticGroup group="224.1.1.1" source="0.0.0.0"/>
        <igmpSnoopStaticGroup group="225.1.1.1" source="2.2.2.2"/>
      </fvRsPathAtt>
    </fvAEPg>
  </fvAp>
</fvTenant>
```

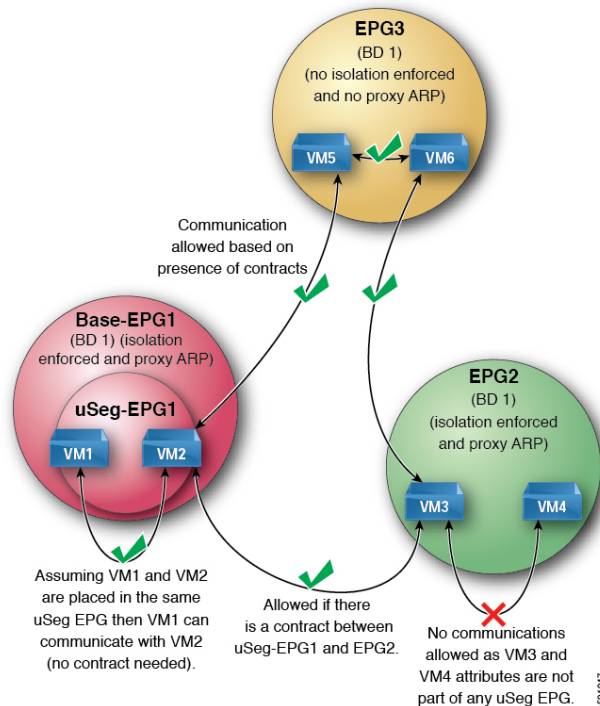
Proxy ARP

About Proxy ARP

Proxy ARP in Cisco ACI enables endpoints within a network or subnet to communicate with other endpoints without knowing the real MAC address of the endpoints. Proxy ARP is aware of the location of the traffic destination, and offers its own MAC address as the final destination instead.

To enable Proxy ARP, intra-EPG endpoint isolation must be enabled on the EPG see the following figure for details. For more information about intra-EPG isolation and Cisco ACI, see the *Cisco ACI Virtualization Guide*.

Figure 25: Proxy ARP and Cisco APIC



Proxy ARP within the Cisco ACI fabric is different from the traditional proxy ARP. As an example of the communication process, when proxy ARP is enabled on an EPG, if an endpoint A sends an ARP request for endpoint B and if endpoint B is learned within the fabric, then endpoint A will receive a proxy ARP response from the bridge domain (BD) MAC. If endpoint A sends an ARP request for endpoint B, and if endpoint B is not learned within the ACI fabric already, then the fabric will send a proxy ARP request within the BD. Endpoint B will respond to this proxy ARP request back to the fabric. At this point, the fabric does not send a proxy ARP response to endpoint A, but endpoint B is learned within the fabric. If endpoint A sends another ARP request to endpoint B, then the fabric will send a proxy ARP response from the BD MAC.

The following example describes the proxy ARP resolution steps for communication between clients VM1 and VM2:

1. VM1 to VM2 communication is desired.

Figure 26: VM1 to VM2 Communication is Desired.

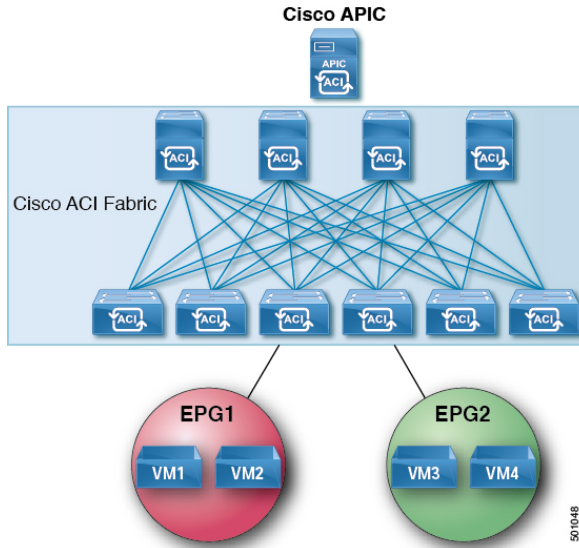


Table 4: ARP Table State

Device	State
VM1	IP = * MAC = *
ACI fabric	IP = * MAC = *
VM2	IP = * MAC = *

- VM1 sends an ARP request with a broadcast MAC address to VM2.

Figure 27: VM1 sends an ARP Request with a Broadcast MAC address to VM2

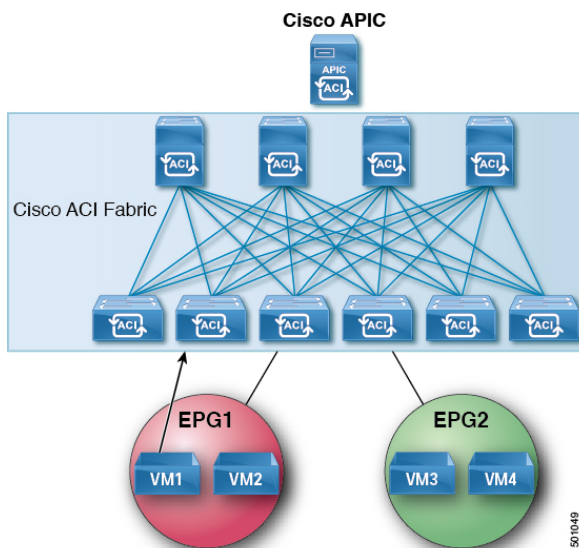


Table 5: ARP Table State

Device	State
VM1	IP = VM2 IP; MAC = ?
ACI fabric	IP = VM1 IP; MAC = VM1 MAC
VM2	IP = * MAC = *

- The ACI fabric floods the proxy ARP request within the bridge domain (BD).

Figure 28: ACI Fabric Floods the Proxy ARP Request within the BD

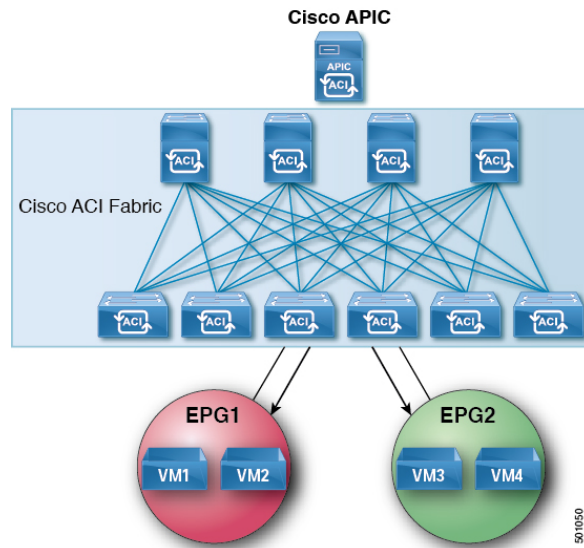


Table 6: ARP Table State

Device	State
VM1	IP = VM2 IP; MAC = ?
ACI fabric	IP = VM1 IP; MAC = VM1 MAC
VM2	IP = VM1 IP; MAC = BD MAC

- VM2 sends an ARP response to the ACI fabric.

Figure 29: VM2 Sends an ARP Response to the ACI Fabric

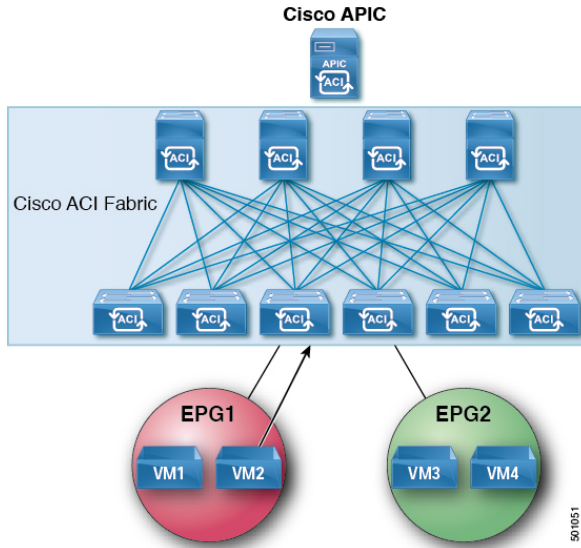


Table 7: ARP Table State

Device	State
VM1	IP = VM2 IP; MAC = ?
ACI fabric	IP = VM1 IP; MAC = VM1 MAC
VM2	IP = VM1 IP; MAC = BD MAC

- 5. VM2 is learned.

Figure 30: VM2 is Learned

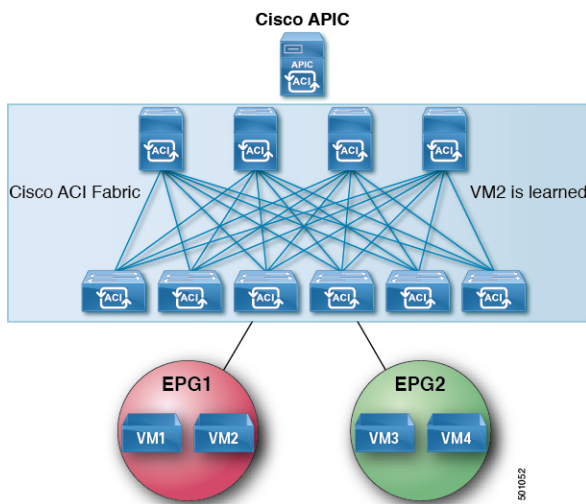


Table 8: ARP Table State

Device	State
VM1	IP = VM2 IP; MAC = ?
ACI fabric	IP = VM1 IP; MAC = VM1 MAC IP = VM2 IP; MAC = VM2 MAC
VM2	IP = VM1 IP; MAC = BD MAC

- VM1 sends an ARP request with a broadcast MAC address to VM2.

Figure 31: VM1 Sends an ARP Request with a Broadcast MAC Address to VM2

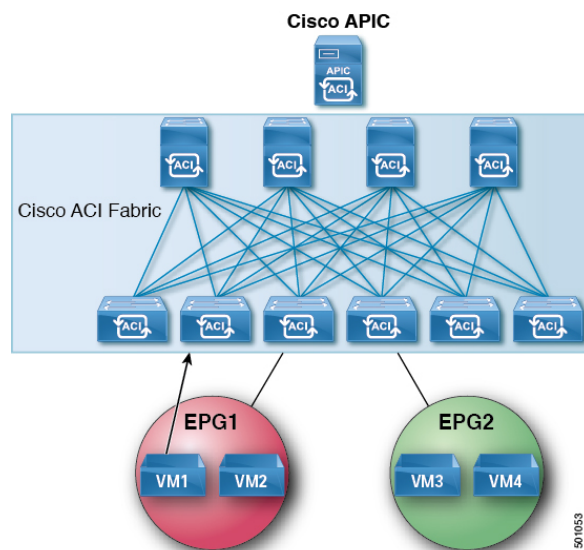


Table 9: ARP Table State

Device	State
VM1	IP = VM2 IP; MAC = ?
ACI fabric	IP = VM1 IP; MAC = VM1 MAC IP = VM2 IP; MAC = VM2 MAC
VM2	IP = VM1 IP; MAC = BD MAC

- The ACI fabric sends a proxy ARP response to VM1.

Figure 32: ACI Fabric Sends a Proxy ARP Response to VM1

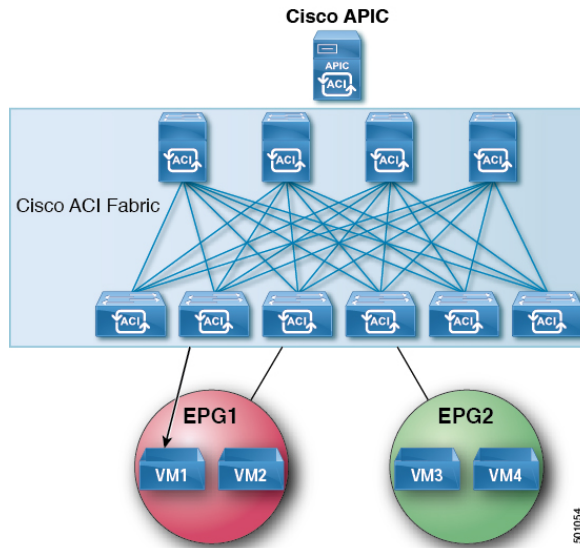


Table 10: ARP Table State

Device	State
VM1	IP = VM2 IP; MAC = BD MAC
ACI fabric	IP = VM1 IP; MAC = VM1 MAC IP = VM2 IP; MAC = VM2 MAC
VM2	IP = VM1 IP; MAC = BD MAC

Guidelines and Limitations

Consider these guidelines and limitations when using Proxy ARP:

- Proxy ARP is supported only on isolated EPGs. If an EPG is not isolated, a fault will be raised. For communication to happen within isolated EPGs with proxy ARP enabled, you must configure uSeg EPGs. For example, within the isolated EPG, there could be multiple VMs with different IP addresses, and you can configure a uSeg EPG with IP attributes matching the IP address range of these VMs.
- ARP requests from isolated endpoints to regular endpoints and from regular endpoints to isolated endpoints do not use proxy ARP. In such cases, endpoints communicate using the real MAC addresses of destination VMs.

Configuring Proxy ARP Using the REST API

Before you begin

- Intra-EPG isolation must be enabled on the EPG where proxy ARP has to be enabled.

Configure proxy ARP.

Example:

```
<polUni>
  <fvTenant name="Tenant1" status="">
    <fvCtx name="EngNet"/>
    <!-- bridge domain -->
    <fvBD name="BD1">
      <fvRsCtx tnFvCtxName="EngNet" />
      <fvSubnet ip="1.1.1.1/24"/>
    </fvBD>
    <fvAp name="Tenant1_app">
      <fvAEPg name="Tenant1_epg" pcEnfPref="enforced" fwdCtrl="proxy-arp">
        <fvRsBd tnFvBDName="BD1" />
        <fvRsDomAtt tDn="uni/vmmp-VMware/dom-dom9"/>
      </fvAEPg>
    </fvAp>
  </fvTenant>
</polUni>
```

Flood on Encapsulation

Configuring Flood in Encapsulation for All Protocols and Proxy ARP Across Encapsulations

Cisco Application Centric Infrastructure (ACI) uses the bridge domain as the Layer 2 broadcast boundary. Each bridge domain can include multiple endpoint groups (EPGs), and each EPG can be mapped to multiple virtual or physical domains. Each EPG can also use different VLAN encapsulation pools in each domain. Each EPG can also use different VLAN or VXLAN encapsulation pools in each domain.

Ordinarily, when you put multiple EPGs within bridge domains, broadcast flooding sends traffic to all the EPGs in the bridge domain. Because EPGs are used to group endpoints and manage traffic to fulfill specific functions, sending the same traffic to all the EPGs in the bridge domain is not always practical.

The flood in encapsulation feature helps to consolidate bridge domains in your network. The feature does so by enabling you to control broadcast flooding to endpoints within the bridge domain based on the encapsulation of the virtual or physical domain that the EPGs are associated with.

Flood in encapsulation requires the bridge domain to be configured with a subnet and with IP routing because in order to allow communication between endpoints of different EPGs in the same bridge domain Cisco ACI performs proxy ARP.

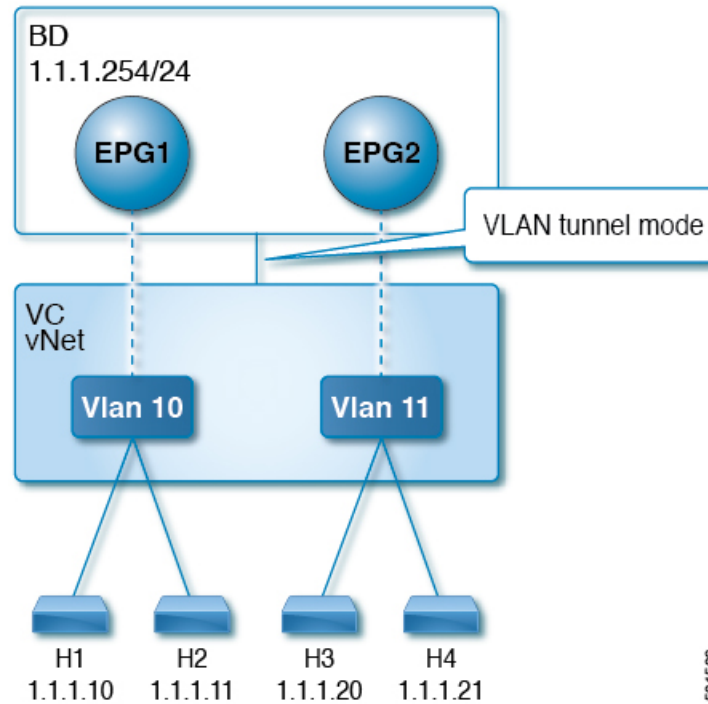
Example of Flood in Encapsulation Use Case with VLAN Encapsulation

Flood in encapsulation is often used when the external device is using Virtual Connect Tunnel mode where one MAC address is maintained per vNet because of VLAN-agnostic MAC learning.

Using multiple VLANs in tunnel mode can introduce a few challenges. In a typical deployment using Cisco ACI with a single tunnel, as illustrated in the following figure, there are multiple EPGs under one bridge

domain. In this case, certain traffic is flooded within the bridge domain (and thus in all the EPGs), with the risk of MAC learning ambiguities that can cause forwarding errors.

Figure 33: Challenges of Cisco ACI with VLAN Tunnel Mode



In this topology, the blade switch (virtual connect in this example) has a single tunnel network defined that uses one uplink to connect with the Cisco ACI leaf node. Two user VLANs, VLAN 10 and VLAN 11 are carried over this link. The bridge domain is set in flooding mode as the servers' gateways are outside the Cisco ACI cloud. ARP negotiations occur in the following process:

- The server sends one ARP broadcast request over the VLAN 10 network.
- The ARP packet travels through the tunnel network to the external server, which records the source MAC address, learned from its downlink.
- The server then forwards the packet out its uplink to the Cisco ACI leaf switch.
- The Cisco ACI fabric sees the ARP broadcast packet entering on access port VLAN 10 and maps it to EPG1.
- Because the bridge domain is set to flood ARP packets, the packet is flooded within the bridge domain and thus to the ports under both EPGs as they are in the same bridge domain.
- The same ARP broadcast packet comes back over the same uplink.
- The blade switch sees the original source MAC address from this uplink.

Result: The blade switch has the same MAC address learned from both the downlink port and uplink port within its single MAC forwarding table, causing traffic disruptions.

Recommended Solution

The flood in encapsulation option is used to limit flooding traffic inside the bridge domain to a single encapsulation. When EPG1/VLAN X and EPG2/VLAN Y share the same bridge domain and flood in encapsulation is enabled, the encapsulation flooding traffic does not reach the other EPG/VLAN.

Beginning with Cisco Application Policy Infrastructure Controller (APIC) release 3.1(1), on the Cisco Nexus 9000 series switches (with names ending with EX and FX and onwards), all protocols are flooded in encapsulation. Also, when flood in encapsulation is enabled under the bridge domain for any inter-VLAN traffic, Proxy ARP ensures that the MAC flap issue does not occur. It also limits all flooding (ARP, GARP, and BUM) to the encapsulation. The restriction applies for all EPGs under the bridge domain where it is enabled.

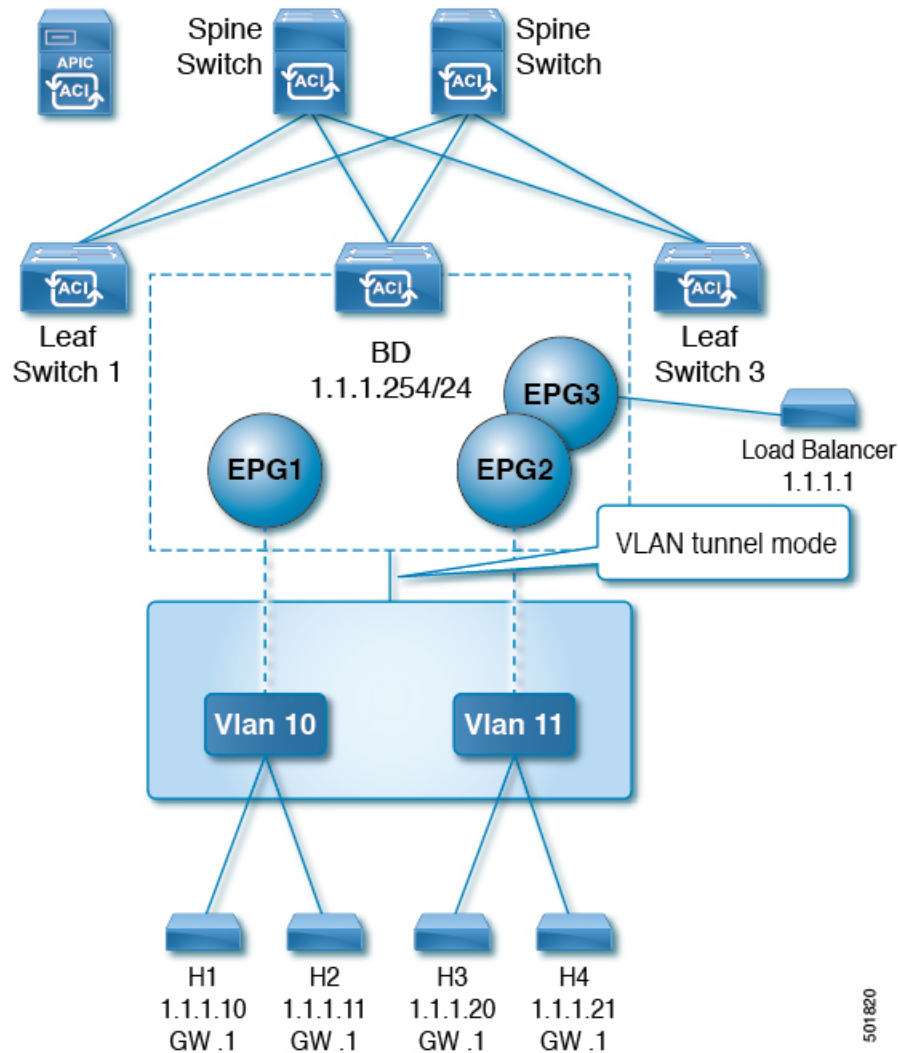


Note Before Cisco APIC release 3.1(1), these features are not supported (proxy ARP and all protocols being included when flooding within encapsulation). In an earlier Cisco APIC release or earlier generation switches (without EX or FX on their names), if you enable flood in encapsulation it does not function, no informational fault is generated, but Cisco APIC decreases the health score by 1.

Beginning with Cisco APIC release 3.2(5), you can configure flood in encapsulation for EPGs associated with VXLAN encapsulation. Previously, only VLANs were supported for flood in encapsulation for virtual domains. You configure flood in encapsulation when you create or modify a bridge domain or an EPG.

The recommended solution is to support multiple EPGs under one bridge domain by adding an external switch. This design with multiple EPGs under one bridge domain with an external switch is illustrated in the following figure.

Figure 34: Design with Multiple EPGs Under one Bridge Domain with an External Switch



Within the same bridge domain, some EPGs can be service nodes and other EPGs can have flood in encapsulation configured. A load balancer resides on a different EPG. The load balancer receives packets from the EPGs and sends them to the other EPGs; there is no Proxy ARP and flood within encapsulation does not take place.

Multi-Destination Protocol Traffic

The EPG/bridge domain level broadcast segmentation is supported for the following network control protocols:

- OSPF
- EIGRP
- LACP
- IS-IS
- BGP

- IGMP
- PIM
- STP-BPDU (flooded within EPG)
- ARP/GARP (controlled by ARP Proxy)
- ND

Flood in Encapsulation Limitations

The following limitations apply when using flood in encapsulation for all protocols:

- Flood in encapsulation does not work in ARP unicast mode.
- Neighbor Solicitation (Proxy NS/ND) is not supported for this release.
- Because proxy Address Resolution Protocol (ARP) is enabled implicitly, ARP traffic can go to the CPU for communication between different encapsulations.
To ensure even distribution to different ports to process ARP traffic, enable per-port Control Plane Policing (CoPP) for ARP with flood in encapsulation.
- Flood in encapsulation is supported only in bridge domain in flood mode and ARP in flood mode. Bridge domain spine proxy mode is not supported.
- IPv4 Layer 3 multicast is not supported.
- IPv6 NS/ND proxy is not supported when flood in encapsulation is enabled. As a result, the connection between two endpoints that are under same IPv6 subnet but resident in EPGs with different encapsulation may not work.
- Virtual machine migration to a different VLAN has momentary issues (60 seconds).
- Setting up communication between virtual machines through a firewall, as a gateway, is not recommended because if the virtual machine IP address changes to the gateway IP address instead of the firewall IP address, then the firewall can be bypassed.
- Prior releases are not supported (even interoperating between prior and current releases).
- A mixed-mode topology with older-generation Application Leaf Engine (ALE) and Application Spine Engine (ASE) is not recommended and is not supported with flood in encapsulation. Enabling them together can prevent QoS priorities from being enforced.
- Flood in encapsulation is not supported with Remote Leaf switches and Cisco ACI Multi-Site.
- Flood in encapsulation is not supported for Common Pervasive Gateway. See the chapter "Common Pervasive Gateway" in the [Cisco APIC Layer 3 Networking Configuration Guide](#).
- Flood in encapsulation is not supported on EPGs where microsegmentation is configured.
- If you configure the flood in encapsulation on all EPGs of a bridge domain, ensure that you configure the flood in encapsulation on the bridge domain as well.
- IGMP snooping is not supported with flood in encapsulation.
- There is a condition that causes Cisco ACI to flood in the bridge domain (instead of the encapsulation) packets that are received on an EPG that is configured for flood in encapsulation. This happens regardless

of whether the administrator configured flood in encapsulation directly on the EPG or on the bridge domain. The condition for this forwarding behavior is if the ingress leaf node has a remote endpoint for the destination MAC address while the egress leaf node does not have a corresponding local endpoint. This can happen due to reasons such as an interface flapping, an endpoint flush due to STP TCN, learning being disabled on the bridge domain due to an excessive amount of moves, and so on.

- A Layer 3 gateway must be in the Cisco ACI fabric.

Configuring Flood on Encapsulation Using the REST API

Configure flood on encapsulation using the REST API.

Enable flood on encapsulation.

To enable flood on encapsulation, send a post with XML such as the following:

Example:

```
<fvAEPg prio="unspecified" prefGrMemb="exclude" pcEnfPref="unenforced" nameAlias="" name="epg900"
matchT="AtleastOne" isAttrBasedEPg="no" fwdCtrl="" floodOnEncap="enabled"
dn="uni/tn-coke/ap-customer/epg-epg900" descr="">
</fvAEPg>
```

MACsec

About MACsec

MACsec is an IEEE 802.1AE standards based Layer 2 hop-by-hop encryption that provides data confidentiality and integrity for media access independent protocols.

MACsec, provides MAC-layer encryption over wired networks by using out-of-band methods for encryption keying. The MACsec Key Agreement (MKA) Protocol provides the required session keys and manages the required encryption keys.

The 802.1AE encryption with MKA is supported on all types of links, that is, host facing links (links between network access devices and endpoint devices such as a PC or IP phone), or links connected to other switches or routers.

MACsec encrypts the entire data except for the Source and Destination MAC addresses of an Ethernet packet. The user also has the option to skip encryption up to 50 bytes after the source and destination MAC address.

To provide MACsec services over the WAN or Metro Ethernet, service providers offer Layer 2 transparent services such as E-Line or E-LAN using various transport layer protocols such as Ethernet over Multiprotocol Label Switching (EoMPLS) and L2TPv3.

The packet body in an EAP-over-LAN (EAPOL) Protocol Data Unit (PDU) is referred to as a MACsec Key Agreement PDU (MKPDU). When no MKPDU is received from a participant after 3 hearbeats (each heartbeat is of 2 seconds), peers are deleted from the live peer list. For example, if a client disconnects, the participant

on the switch continues to operate MKA until 3 heartbeats have elapsed after the last MKPDU is received from the client.

APIC Fabric MACsec

The APIC will be responsible for the MACsec keychain distribution to all the nodes in a Pod or to particular ports on a node. Below are the supported MACsec keychain and MACsec policy distribution supported by the APIC.

- A single user provided keychain and policy per Pod
- User provided keychain and user provided policy per fabric interface
- Auto generated keychain and user provided policy per Pod

A node can have multiple policies deployed for more than one fabric link. When this happens, the per fabric interface keychain and policy are given preference on the affected interface. The auto generated keychain and associated MACsec policy are then given the least preference.

APIC MACsec supports two security modes. The MACsec **must secure** only allows encrypted traffic on the link while the **should secure** allows both clear and encrypted traffic on the link. Before deploying MACsec in **must secure** mode, the keychain must be deployed on the affected links or the links will go down. For example, a port can turn on MACsec in **must secure** mode before its peer has received its keychain resulting in the link going down. To address this issue the recommendation is to deploy MACsec in **should secure** mode and once all the links are up then change the security mode to **must secure**.



Note Any MACsec interface configuration change will result in packet drops.

MACsec policy definition consists of configuration specific to keychain definition and configuration related to feature functionality. The keychain definition and feature functionality definitions are placed in separate policies. Enabling MACsec per Pod or per interface involves deploying a combination of a keychain policy and MACsec functionality policy.



Note Using internal generated keychains do not require the user to specify a keychain.

APIC Access MACsec

MACsec is used to secure links between leaf switch L3out interfaces and external devices. APIC provides GUI and CLI to allow users to program the MACsec keys and MacSec configuration for the L3Out interfaces on the fabric on a per physical/pc/vpc interface basis. It is the responsibility of the user to make sure that the external peer devices are programmed with the correct MacSec information.

Guidelines and Limitations for MACsec

Configure MACsec according to the following guidelines and limitations:

- MACsec is supported on the following switches:
 - N9K-C93108TC-FX

- N9K-C93180YC-FX
 - N9K-C93216TC-FX2
 - N9K-C93240YC-FX2
 - N9K-C9332C
 - N9K-C93360YC-FX2
 - N9K-C9336C-FX2
 - N9K-C9348GC-FXP, only with 10G+
 - N9K-C9364C
- MACsec is supported on the following line card:
 - N9K-X9736C-FX
 - MACsec is not supported on 10G QSA modules.
 - Beginning with Cisco Application Policy Infrastructure Controller (APIC) release 4.0, MACsec is supported on remote leaf switches.
 - FEX ports are not supported for MACsec.
 - The **must-secure** mode is not supported at the pod level.
 - A MACsec policy with the name "default" is not supported.
 - Auto key generation is only supported at the pod level for fabric ports.
 - Do not clean reboot a node if the fabric ports of that node is running MACsec in **must-secure** mode.
 - Adding a new node to a pod or stateless reboot of a node in a pod that is running MACsec, **must-secure** mode requires changing the mode to **should-secure** for the node to join the pod.
 - Only initiate an upgrade or downgrade if the fabric links are in the **should-secure** mode. After the upgrade or downgrade has completed, you can change the mode to **must-secure**. Upgrading or downgrading in the **must-secure** mode results in nodes losing connectivity to the fabric. Recovering from connectivity loss requires you to configure in **should-secure** mode the fabric links of the nodes that are visible to the Cisco APIC. If the fabric was downgraded to a version which does not support MACsec, then nodes which are out of fabric will need to be clean rebooted.
 - For a PC or vPC interface, MACsec can be deployed using policy groups per PC or vPC interface. Port selectors are used to deploy the policies to a particular set of ports. Therefore, you must create the correct port selector that corresponds to the L3Out interfaces.
 - We recommend that you configure MACsec polices with the **should-secure** mode before you export a configuration.
 - All of the links on a spine switch are considered to be fabric links. However, if a spine switch link is used for IPN connectivity, then this link will be treated as an access link. This means that a MACsec access policy must be used to deploy MACsec on these links.
 - If a remote leaf fabric link is used for IPN connectivity, then this link will be treated as an access link. A MACsec access policy needs to be used to deploy MACsec on these links.

- Improper deployment of **must-secure** mode on remote leaf switch fabric links can result in loss of connectivity to the fabric. Follow the instructions provided in [Deploying must-secure mode, on page 300](#) to prevent such issues.
- MACsec sessions can take up to a minute to form or tear down when a new key is added to an empty keychain or an active key is deleted from a keychain.
- Before reloading a line card or fabric module on a spine switch, all **must-secure** links should be changed to the **should-secure** mode. After the reload completes and the session comes up in the **should-secure** mode, change the mode to **must-secure**.
- When selecting the cipher suite AES 128 or AES 256 without Extended Packet Numbering (XPN), you must explicitly specify the Security Association Key (SAK) expiry time. Leaving the SAK expiry time value at the default ("disabled") can cause interfaces to go out of service randomly.
- A replay window is necessary to support the use of MACsec over provider networks that reorder frames. Frames within the window can be received out of order, but are not replay protected. The default window size is 64. The replay window size can be configured in the range of 0 to $2^{32}-1$ if you use the Cisco APIC GUI or CLI. If you use a XPN cipher suite, the maximum replay window size is $2^{30}-1$, and if you configure a higher window size, the window size gets restricted to $2^{30}-1$. If you change the cipher suite to a non-XPN cipher suite, then there is no restriction and the configured window size is used.
- Link-level flow control (LLFC) and priority flow control (PFC) are not supported with MACsec.
- Cisco APIC does not support passing MACsec through its infrastructure for clients.

Deploying must-secure mode

Incorrect deployment procedure of a policy that is configured for **must-secure** mode can result in a loss of connectivity. The procedure below should be followed in order to prevent such issues:

- It is necessary to ensure that each link pair has their keychains before enabling MACsec **must-secure** mode. To ensure this, the recommendation is to deploy the policy in **should-secure** mode, and once MACsec sessions are active on the expected links, change the mode to **must-secure**.
- Attempting to replace the keychain on a MACsec policy that is configured to **must-secure** can cause links to go down. The recommended procedure outlined below should be followed in this case:
 - Change MACsec policy that is using the new keychain to **should-secure** mode.
 - Verify that the affected interfaces are using should-secure mode.
 - Update MACsec policy to use new keychain.
 - Verify that relevant interfaces with active MACsec sessions are using the new keychain.
 - Change MACsec policy to **must-secure** mode.
- The following procedure should be followed to disable/remove a MACsec policy deployed in must-secure mode:
 - Change the MACsec policy to **should-secure**.
 - Verify that the affected interfaces are using **should-secure** mode.
 - Disable/remove the MACsec policy.

Keychain Definition

- There should be one key in the keychain with a start time of **now**. If **must-secure** is deployed with a keychain that doesn't have a key that is immediately active then traffic will be blocked on that link until the key becomes current and a MACsec session is started. If **should-secure** mode is being used then traffic will be unencrypted until the key becomes current and a MACsec session has started.
- There should be one key in the keychain with an end time of **infinite**. When a keychain expires, then traffic is blocked on affected interfaces which are configured for **must-secure** mode. Interfaces configured for **should-secure** mode transmit unencrypted traffic.
- There should be overlaps in the end time and start time of keys that are used sequentially to ensure the MACsec session stays up when there is a transition between keys.

Configuring MACsec Using the REST API

Apply a MACsec fabric policy to all Pods in the fabric:

Example:

```
<fabricInst>
  <macsecFabPolCont>
    <macsecFabParamPol name="fabricParam1" secPolicy="should-secure" replayWindow="120"
  >
    </macsecFabParamPol>
    <macsecKeyChainPol name="fabricKC1">
      <macsecKeyPol name="Key1"
preSharedKey="0102030405060708090A0B0C0D0E0F100102030405060708090A0B0C0D0E0F10"
keyName="A1A2A3A0" startTime="now" endTime="infinite"/>
      </macsecKeyChainPol>
    </macsecFabPolCont>

    <macsecFabIfPol name="fabricPodPol1" useAutoKeys="0">
      <macsecRsToParamPol tDn="uni/fabric/macsecpcontfab/fabparam-fabricParam1"/>
      <macsecRsToKeyChainPol tDn="uni/fabric/macsecpcontfab/keychain-fabricKC1"/>
    </macsecFabIfPol>

    <fabricFuncP>
    <fabricPodPGrp name = "PodPG1">
    <fabricRsMacsecPol tnMacsecFabIfPolName="fabricPodPol1"/>
      </fabricPodPGrp>
    </fabricFuncP>

    <fabricPodP name="PodP1">
    <fabricPodS name="pod1" type="ALL">
    <fabricRsPodPGrp tDn="uni/fabric/funcprof/podpgrp-PodPG1"/>
      </fabricPodS>
    </fabricPodP>

  </fabricInst>
```

Applying a MACsec access policy on eth1/4 of leaf-101:

Example:

```
<infraInfra>
  <macsecPolCont>
    <macsecParamPol name="accessParam1" secPolicy="should-secure" replayWindow="120"
  >
    </macsecParamPol>
    <macsecKeyChainPol name="accessKC1">
```

```

        <macsecKeyPol name="Key1"
preSharedKey="0102030405060708090A0B0C0D0E0F100102030405060708090A0B0C0D0E0F10"
keyName="A1A2A3A0" startTime="now" endTime="infinite"/>
        </macsecKeyChainPol>
    </macsecPolCont>

    <macsecIfPol name="accessPol1">
        <macsecRsToParamPol tDn="uni/infra/macsecpcont/paramp-accessParam1"/>
        <macsecRsToKeyChainPol tDn="uni/infra/macsecpcont/keychainp-accessKC1"/>
    </macsecIfPol>

    <infraFuncP>
    <infraAccPortGrp name = "LeTestPGrp">
    <infraRsMacsecIfPol tnMacsecIfPolName="accessPol1"/>
    </infraAccPortGrp>
    </infraFuncP>

    <infraHPathS name="leaf">
    <infraRsHPathAtt tDn="topology/pod-1/paths-101/pathep-[eth1/4]" />
    <infraRsPathToAccBaseGrp tDn="uni/infra/funcprof/accportgrp-LeTestPGrp" />
    </infraHPathS>

</infraInfra>

```

Applying a MACsec fabric policy on eth1/49 of leaf-101 and eth 5/1 of spine-102:

```

<fabricInst>
    <macsecFabPolCont>
        <macsecFabParamPol name="fabricParam1" secPolicy="should-secure" replayWindow="120"
    >
        </macsecFabParamPol>
        <macsecKeyChainPol name="fabricKC1">
            <macsecKeyPol name="Key1"
preSharedKey="0102030405060708090A0B0C0D0E0F100102030405060708090A0B0C0D0E0F10"
keyName="A1A2A3A0" startTime="now" endTime="infinite"/>
            </macsecKeyChainPol>
        </macsecFabPolCont>

        <macsecFabIfPol name="fabricPol1" useAutoKeys="0">
            <macsecRsToParamPol tDn="uni/fabric/macsecpcontfab/fabparamp-fabricParam1"/>
            <macsecRsToKeyChainPol tDn="uni/fabric/macsecpcontfab/keychainp-fabricKC1"/>
        </macsecFabIfPol>

        <fabricFuncP>
        <fabricLePortPGrp name = "LeTestPGrp">
        <fabricRsMacsecFabIfPol tnMacsecFabIfPolName="fabricPol1"/>
        </fabricLePortPGrp>

        <fabricSpPortPGrp name = "SpTestPGrp">
        <fabricRsMacsecFabIfPol tnMacsecFabIfPolName="fabricPol1"/>
        </fabricSpPortPGrp>
    </fabricFuncP>

    <fabricLFPathS name="leaf">
    <fabricRsLFPathAtt tDn="topology/pod-1/paths-101/pathep-[eth1/49]" />
    <fabricRsPathToLePortPGrp tDn="uni/fabric/funcprof/leportgrp-LeTestPGrp" />
    </fabricLFPathS>

    <fabricSpPortP name="spine_profile">
    <fabricSFPortS name="spineIf" type="range">
        <fabricPortBlk name="spBlk" fromCard="5" fromPort="1" toCard="5" toPort="1" />
        <fabricRsSpPortPGrp tDn="uni/fabric/funcprof/spportgrp-SpTestPGrp" />
    </fabricSFPortS>
    </fabricSpPortP>

```



```
<fabricSpineP name="SpNode" >  
  <fabricRsSpPortP tDn="uni/fabric/sportp-spine_profile" />  
  <fabricSpineS name="spsw" type="range">  
    <fabricNodeBlk name="node102" to_="102" from_="102" />  
  </fabricSpineS>  
</fabricSpineP>  
</fabricInst>
```




CHAPTER 15

Provisioning Layer 3 Outside Connections

- [Layer 3 Outside Connections](#), on page 305
- [Layer 3 Routed and Sub-Interface Port Channels](#), on page 315
- [Cisco ACI GOLF](#), on page 320
- [Multipod](#), on page 328
- [Anycast Services](#), on page 333
- [Remote Leaf Switches](#), on page 335
- [HSRP](#), on page 347
- [IP Multicast](#), on page 351
- [Pervasive Gateway](#), on page 357
- [Explicit Prefix Lists](#), on page 358
- [IP Address Aging Tracking](#), on page 366
- [Route Summarization](#), on page 367
- [Route Controls](#), on page 371
- [Layer 3 to Layer 3 Out Inter-VRF Leaking](#), on page 372
- [Overview Interleak Redistribution for MP-BGP](#), on page 374
- [Configuring Interleak of External Routes Using the REST API](#), on page 374
- [SVI External Encapsulation Scope](#), on page 375
- [SVI Auto State](#), on page 378
- [Routing Protocols](#), on page 380
- [Neighbor Discovery](#), on page 398
- [Microsoft NLB](#), on page 402
- [MLD Snooping](#), on page 403

Layer 3 Outside Connections

Configuring a Tenant Layer 3 Outside Network Connection Overview

This topic provides a typical example of how to configure a Layer 3 Outside for tenant networks when using Cisco APIC.



Note Cisco ACI does not support IP fragmentation. Therefore, when you configure Layer 3 Outside (L3Out) connections to external routers, or Multi-Pod connections through an Inter-Pod Network (IPN), it is recommended that the interface MTU is set appropriately on both ends of a link. On some platforms, such as Cisco ACI, Cisco NX-OS, and Cisco IOS, the configurable MTU value does not take into account the Ethernet headers (matching IP MTU, and excluding the 14-18 Ethernet header size), while other platforms, such as IOS-XR, include the Ethernet header in the configured MTU value. A configured value of 9000 results in a max IP packet size of 9000 bytes in Cisco ACI, Cisco NX-OS, and Cisco IOS, but results in a max IP packet size of 8986 bytes for an IOS-XR untagged interface.

For the appropriate MTU values for each platform, see the relevant configuration guides.

We highly recommend that you test the MTU using CLI-based commands. For example, on the Cisco NX-OS CLI, use a command such as `ping 1.1.1.1 df-bit packet-size 9000 source-interface ethernet 1/1`.

Configuring Layer 3 Outside for Tenant Networks Using the REST API

The external routed network that is configured in the example can also be extended to support both IPv4 and IPv6. Both IPv4 and IPv6 routes can be advertised to and learned from the external routed network. To configure an L3Out for a tenant network, send a post with XML such as the example.

This example is broken into steps for clarity. For a merged example, see [REST API Example: L3Out, on page 312](#).

Before you begin

- Configure the node, port, functional profile, AEP, and Layer 3 domain.
- Create the external routed domain and associate it to the interface for the L3Out.
- Configure a BGP route reflector policy to propagate the routes within the fabric.

For an XML example of these prerequisites, see [REST API Example: L3Out Prerequisites, on page 311](#).

Step 1 Configure the tenant, VRF, and bridge domain.

This example configures tenant `t1` with VRF `v1` and bridge domain `bd1`. The tenant, VRF, and BD are not yet deployed.

Example:

```
<fvTenant name="t1">
  <fvCtx name="v1"/>
  <fvBD name="bd1">
    <fvRsCtx tnFvCtxName="v1"/>
    <fvSubnet ip="44.44.44.1/24" scope="public"/>
    <fvRsBDToOut tnL3extOutName="l3out1"/>
  </fvBD/>
</fvTenant>
```

Step 2 Configure an application profile and application EPG.

This example configures application profile `app1` (on node 101), EPG `epg1`, and associates the EPG with `bd1` and the contract `httpCtct`, as the consumer.

Example:

```

<fvAp name="appl">
  <fvAEPg name="epg1">
    <fvRsDomAtt instrImedcy="immediate" tDn="uni/phys-dom1"/>
    <fvRsBd tnFvBDName="bd1" />
    <fvRsPathAtt encap="vlan-2011" instrImedcy="immediate" mode="regular"
tDn="topology/pod-1/paths-101/pathep-[eth1/3]"/>
    <fvRsCons tnVzBrCPName="httpCtrct"/>
  </fvAEPg>
</fvAp>

```

Step 3 Configure the node and interface.

This example configures VRF `v1` on node 103 (the border leaf switch), with the node profile, `nodep1`, and router ID 11.11.11.103. It also configures interface `eth1/3` as a routed interface (Layer 3 port), with IP address 12.12.12.1/24 and Layer 3 domain `dom1`.

Example:

```

<l3extOut name="l3out1">
  <l3extRsEctx tnFvCtxName="v1"/>
  <l3extLNodeP name="nodep1">
    <l3extRsNodeL3OutAtt rtrId="11.11.11.103" tDn="topology/pod-1/node-103"/>
    <l3extLIIfP name="ifp1"/>
    <l3extRsPathL3OutAtt addr="12.12.12.3/24" ifInstT="l3-port"
tDn="topology/pod-1/paths-103/pathep-[eth1/3]"/>
    </l3extLIIfP>
  </l3extLNodeP>
  <l3extRsL3DomAtt tDn="uni/l3dom-dom1"/>
</l3extOut>

```

Step 4 Configure the routing protocol.

This example configures BGP as the primary routing protocol, with a BGP peer with the IP address, 15.15.15.2 and ASN 100.

Example:

```

<l3extOut name="l3out1">
  <l3extLNodeP name="nodep1">
    <bgpPeerP addr="15.15.15.2">
      <bgpAsP asn="100"/>
    </bgpPeerP>
  </l3extLNodeP>
  <bgpExtP/>
</l3extOut>

```

Step 5 Configure the connectivity routing protocol.

This example configures OSPF as the communication protocol, with regular area ID 0.0.0.0.

Example:

```

<l3extOut name="l3out1">
  <ospfExtP areaId="0.0.0.0" areaType="regular"/>
  <l3extLNodeP name="nodep1">
    <l3extLIIfP name="ifp1">
      <ospfIIfP/>
    </l3extLIIfP>
  </l3extLNodeP>
</l3extOut>

```

Step 6 Configure the external EPG.

This example configures the network 20.20.20.0/24 as external network `extnw1`. It also associates `extnw1` with the route control profile `rp1` and the contract `httpCtrct`, as the provider.

Example:

```
<l3extOut name="l3out1">
  <l3extInstP name="extnw1">
    <l3extSubnet ip="20.20.20.0/24" scope="import-security"/>
    <fvRsProv tnVzBrCPName="httpCtrct"/>
  </l3extInstP>
</l3extOut>
```

Step 7 Optional. Configure a route map.

This example configures a route map for the BGP peer in the outbound direction. The route map is applied for routes that match a destination of 200.3.2.0/24. Also, on a successful match (if the route matches this range) the route AS PATH attribute is updated to 200 and 100.

Example:

```
<fvTenant name="t1">
  <rtctrlSubjP name="match-rule1">
    <rtctrlMatchRtDest ip="200.3.2.0/24"/>
  </rtctrlSubjP>
  <l3extOut name="l3out1">
    <rtctrlProfile name="rp1">
      <rtctrlCtxP name="ctxp1" action="permit" order="0">
        <rtctrlScope>
          <rtctrlRsScopeToAttrP tnRtctrlAttrPName="attrp1"/>
        </rtctrlScope>
        <rtctrlRsCtxPToSubjP tnRtctrlSubjPName="match-rule1"/>
      </rtctrlCtxP>
    </rtctrlProfile>
    <l3extInstP name="extnw1">
      <l3extSubnet ip="20.20.20.0/24" scope="import-security"/>
      <l3extRsInstPToProfile direction='export' tnRtctrlProfileName="rp1"/>
      <fvRsProv tnVzBrCPName="httpCtrct"/>
    </l3extInstP>
  </l3extOut>
</fvTenant>
```

Step 8 This example creates filters and contracts to enable the EPGs to communicate. The external EPG and the application EPG are already associated with the contract `httpCtrct` as provider and consumer respectively. The scope of the contract (where it is applied) can be within the application profile, the tenant, the VRF, or it can be used globally (throughout the fabric). In this example, the scope is the VRF (`context`).**Example:**

```
<vzFilter name="http-filter">
  <vzEntry name="http-e" etherT="ip" prot="tcp"/>
</vzFilter>
<vzBrCP name="httpCtrct" scope="context">
  <vzSubj name="subj1">
    <vzRsSubjFiltAtt tnVzFilterName="http-filter"/>
  </vzSubj>
</vzBrCP>
```

Step 9 Configure Advertise Host Routes.**Example:**

```
"<fvBD dn="uni/tn-t1/BD-b100" hostBasedRouting="yes"/>"
```

Configuring Layer 3 Outside for Tenant Networks Using the REST API

The external routed network that is configured in the example can also be extended to support both IPv4 and IPv6. Both IPv4 and IPv6 routes can be advertised to and learned from the external routed network. To configure an L3Out for a tenant network, send a post with XML such as the example.

This example is broken into steps for clarity. For a merged example, see [REST API Example: L3Out, on page 312](#).

Before you begin

- Configure the node, port, functional profile, AEP, and Layer 3 domain.
- Create the external routed domain and associate it to the interface for the L3Out.
- Configure a BGP route reflector policy to propagate the routes within the fabric.

For an XML example of these prerequisites, see [REST API Example: L3Out Prerequisites, on page 311](#).

Step 1 Configure the tenant, VRF, and bridge domain.

This example configures tenant `t1` with VRF `v1` and bridge domain `bd1`. The tenant, VRF, and BD are not yet deployed.

Example:

```
<fvTenant name="t1">
  <fvCtx name="v1"/>
  <fvBD name="bd1">
    <fvRsCtx tnFvCtxName="v1"/>
    <fvSubnet ip="44.44.44.1/24" scope="public"/>
    <fvRsBDToOut tnL3extOutName="l3out1"/>
  </fvBD></>
</fvTenant>
```

Step 2 Configure an application profile and application EPG.

This example configures application profile `app1` (on node 101), EPG `epg1`, and associates the EPG with `bd1` and the contract `httpCtrct`, as the consumer.

Example:

```
<fvAp name="app1">
  <fvAEPg name="epg1">
    <fvRsDomAtt instrImedcy="immediate" tDn="uni/phys-dom1"/>
    <fvRsBd tnFvBDName="bd1" />
    <fvRsPathAtt encap="vlan-2011" instrImedcy="immediate" mode="regular"
tDn="topology/pod-1/paths-101/pathep-[eth1/3]"/>
    <fvRsCons tnVzBrCPName="httpCtrct"/>
  </fvAEPg>
</fvAp>
```

Step 3 Configure the node and interface.

This example configures VRF `v1` on node 103 (the border leaf switch), with the node profile, `nodep1`, and router ID `11.11.11.103`. It also configures interface `eth1/3` as a routed interface (Layer 3 port), with IP address `12.12.12.1/24` and Layer 3 domain `dom1`.

Example:

```
<l3extOut name="l3out1">
  <l3extRsEctx tnFvCtxName="v1"/>
```

```

    <l3extLNodeP name="nodep1">
      <l3extRsNodeL3OutAtt rtrId="11.11.11.103" tDn="topology/pod-1/node-103"/>
      <l3extLIIfP name="ifp1"/>
      <l3extRsPathL3OutAtt addr="12.12.12.3/24" ifInstT="l3-port"
tDn="topology/pod-1/paths-103/pathep-[eth1/3]"/>
    </l3extLIIfP>
  </l3extLNodeP>
  <l3extRsL3DomAtt tDn="uni/l3dom-dom1"/>
</l3extOut>

```

Step 4 Configure the routing protocol.

This example configures BGP as the primary routing protocol, with a BGP peer with the IP address, 15.15.15.2 and ASN 100.

Example:

```

<l3extOut name="l3out1">
  <l3extLNodeP name="nodep1">
    <bgpPeerP addr="15.15.15.2">
      <bgpAsP asn="100"/>
    </bgpPeerP>
  </l3extLNodeP>
  <bgpExtP/>
</l3extOut>

```

Step 5 Configure the connectivity routing protocol.

This example configures OSPF as the communication protocol, with regular area ID 0.0.0.0.

Example:

```

<l3extOut name="l3out1">
  <ospfExtP areaId="0.0.0.0" areaType="regular"/>
  <l3extLNodeP name="nodep1">
    <l3extLIIfP name="ifp1">
      <ospfIfP/>
    </l3extLIIfP>
  </l3extLNodeP>
</l3extOut>

```

Step 6 Configure the external EPG.

This example configures the network 20.20.20.0/24 as external network `extnw1`. It also associates `extnw1` with the route control profile `rp1` and the contract `httpCtrct`, as the provider.

Example:

```

<l3extOut name="l3out1">
  <l3extInstP name="extnw1">
    <l3extSubnet ip="20.20.20.0/24" scope="import-security"/>
    <fvRsProv tnVzBrCPName="httpCtrct"/>
  </l3extInstP>
</l3extOut>

```

Step 7 Optional. Configure a route map.

This example configures a route map for the BGP peer in the outbound direction. The route map is applied for routes that match a destination of 200.3.2.0/24. Also, on a successful match (if the route matches this range) the route AS PATH attribute is updated to 200 and 100.

Example:

```

<fvTenant name="t1">
  <rtctrlSubjP name="match-rule1">
    <rtctrlMatchRtDest ip="200.3.2.0/24"/>

```



```

</rtctrlSubjP>
<l3extOut name="l3out1">
  <rtctrlProfile name="rp1">
    <rtctrlCtxP name="ctxp1" action="permit" order="0">
      <rtctrlScope>
        <rtctrlRsScopeToAttrP tnRtctrlAttrPName="attrp1"/>
      </rtctrlScope>
      <rtctrlRsCtxPToSubjP tnRtctrlSubjPName="match-rule1"/>
    </rtctrlCtxP>
  </rtctrlProfile>
  <l3extInstP name="extnw1">
    <l3extSubnet ip="20.20.20.0/24" scope="import-security"/>
    <l3extRsInstPToProfile direction='export' tnRtctrlProfileName="rp1"/>
    <fvRsProv tnVzBrCPName="httpCtrct"/>
  </l3extInstP>
</l3extOut>
</fvTenant>

```

Step 8 This example creates filters and contracts to enable the EPGs to communicate. The external EPG and the application EPG are already associated with the contract `httpCtrct` as provider and consumer respectively. The scope of the contract (where it is applied) can be within the application profile, the tenant, the VRF, or it can be used globally (throughout the fabric). In this example, the scope is the VRF (`context`).

Example:

```

<vzFilter name="http-filter">
  <vzEntry name="http-e" etherT="ip" prot="tcp"/>
</vzFilter>
<vzBrCP name="httpCtrct" scope="context">
  <vzSubj name="subj1">
    <vzRsSubjFiltAtt tnVzFilterName="http-filter"/>
  </vzSubj>
</vzBrCP>

```

Step 9 Configure Advertise Host Routes.

Example:

```

"<fvBD dn="uni/tn-t1/BD-b100" hostBasedRouting="yes"/>"

```

REST API Example: L3Out Prerequisites

This example configures the node, port, functional profile, AEP, and Layer 3 domain:

```

<?xml version="1.0" encoding="UTF-8"?>
<!-- api/policymgr/mo/.xml -->
<polUni>
  <infraInfra>
    <!-- Node profile -->
    <infraNodeP name="nodeP1">
      <infraLeafS name="leafS1" type="range">
        <infraNodeBlk name="NodeBlk1" from_"=101" to_"=103" />
      </infraLeafS>
      <infraRsAccPortP tDn="uni/infra/accportprof-PortP1" />
    </infraNodeP>
    <!-- Port profile -->
    <infraAccPortP name="PortP1">
      <!-- 12 regular ports -->
      <infraHPortS name="PortS1" type="range">
        <infraPortBlk name="portBlk1" fromCard="1" toCard="1" fromPort="3"
toPort="32"/>

```

```

        <infraRsAccBaseGrp tDn="uni/infra/funcprof/accportgrp-default" />
    </infraHPortS>
</infraAccPortP>
<!-- Functional profile -->
<infraFuncP>
    <!-- Regular port group -->
    <infraAccPortGrp name="default">
        <infraRsAttEntP tDn="uni/infra/attentp-aeP1" />
    </infraAccPortGrp>
</infraFuncP>
<infraAttEntityP name="aeP1">
    <infraRsDomP tDn="uni/phys-dom1"/>
    <infraRsDomP tDn="uni/l3dom-dom1"/>
</infraAttEntityP>
<fvnsVlanInstP name="vlan-1024-2048" allocMode="static">
    <fvnsEncapBlk name="encap" from="vlan-1024" to="vlan-2048" status="created"/>
</fvnsVlanInstP>
</infraInfra>
<physDomP dn="uni/phys-dom1" name="dom1">
    <infraRsVlanNs tDn="uni/infra/vlanns-[vlan-1024-2048]-static"/>
</physDomP>
<l3extDomP name="dom1">
    <infraRsVlanNs tDn="uni/infra/vlanns-[vlan-1024-2048]-static" />
</l3extDomP>
</polUni>

```

The following example configures the required BGP route reflectors:

```

<!-- Spine switches 104 and 105 are configured as route reflectors -->
<?xml version="1.0" encoding="UTF8"?>
<!-- api/policymgr/mo/.xml -->
<polUni>
    <bgpInstPol name="default">
        <bgpAsP asn="100"/>
        <bgpRRP>
            <bgpRRNodePEp id="104"/>
            <bgpRRNodePEp id="105"/>
        </bgpRRP>
    </bgpInstPol>
<fabricFuncP>
    <fabricPodPGrp name="bgpRRPodGrp1">
        <fabricRsPodPGrpBGPRRP tnBgpInstPolName="default"/>
    </fabricPodPGrp>
</fabricFuncP>
<fabricPodP name="default">
    <fabricPods name="default" type="ALL">
        <fabricRsPodPGrp tDn="uni/fabric/funcprof/podpgrp-bgpRRPodGrp1"/>
    </fabricPods>
</fabricPodP>
</polUni>

```

REST API Example: L3Out

The following example provides a merged version of the steps to configure an L3Out using the REST API.

```

<?xml version="1.0" encoding="UTF8"?>
<!-- api/policymgr/mo/.xml -->
<polUni>
    <fvTenant name="t1">
        <fvCtx name="v1"/>
        <fvBD name="bd1">
            <fvRsCtx tnFvCtxName="v1"/>
            <fvSubnet ip="44.44.44.1/24" scope="public"/>
        </fvBD>
    </fvTenant>
</polUni>

```

```

        <fvRsBDToOut tnL3extOutName="l3out1"/>
    </fvBD>
    <fvAp name="appl">
        <fvAEPg name="epg1">
            <fvRsDomAtt instrImedcy="immediate" tDn="uni/phys-dom1"/>
            <fvRsBd tnFvBDName="bd1" />
            <fvRsPathAtt encap="vlan-2011" instrImedcy="immediate" mode="regular"
tDn="topology/pod-1/paths-101/pathep-[eth1/3]"/>
            <fvRsCons tnVzBrCPName="httpCtrct"/>
        </fvAEPg>
    </fvAp>
    <l3extOut name="l3out1">
        <l3extRsEctx tnFvCtxName="v1"/>
        <l3extLNodeP name="nodep1">
            <l3extRsNodeL3OutAtt rtrId="11.11.11.103" tDn="topology/pod-1/node-103"/>
            <l3extLIIfP name="ifp1">
                <l3extRsPathL3OutAtt addr="12.12.12.3/24" ifInstT="l3-port"
tDn="topology/pod-1/paths-103/pathep-[eth1/3]"/>
            </l3extLIIfP>
            <bgpPeerP addr="15.15.15.2">
                <bgpAsP asn="100"/>
            </bgpPeerP>
        </l3extLNodeP>
        <l3extRsL3DomAtt tDn="uni/l3dom-dom1"/>
        <bgpExtP/>
        <ospfExtP areaId="0.0.0.0" areaType="regular"/>
        <l3extInstP name="extnw1" >
            <l3extSubnet ip="20.20.20.0/24" scope="import-security"/>
            <l3extRsInstPToProfile direction="export" tnRtctrlProfileName="rp1"/>
            <fvRsProv tnVzBrCPName="httpCtrct"/>
        </l3extInstP>
        <rtctrlProfile name="rp1">
            <rtctrlCtxP name="ctxp1" action="permit" order="0">
                <rtctrlScope>
                    <rtctrlRsScopeToAttrP tnRtctrlAttrPName="attrp1"/>
                </rtctrlScope>
                <rtctrlRsCtxPToSubjP tnRtctrlSubjPName="match-rule1"/>
            </rtctrlCtxP>
        </rtctrlProfile>
    </l3extOut>
    <rtctrlSubjP name="match-rule1">
        <rtctrlMatchRtDest ip="200.3.2.0/24"/>
    </rtctrlSubjP>
    <rtctrlAttrP name="attrp1">
        <rtctrlSetASPath criteria="prepend">
            <rtctrlSetASPathASN asn="100" order="2"/>
            <rtctrlSetASPathASN asn="200" order="1"/>
        </rtctrlSetASPath>
    </rtctrlAttrP>
    <vzFilter name='http-filter'>
        <vzEntry name="http-e" etherT="ip" prot="tcp"/>
    </vzFilter>
    <vzBrCP name="httpCtrct" scope="context">
        <vzSubj name="subj1">
            <vzRsSubjFiltAtt tnVzFilterName="http-filter"/>
        </vzSubj>
    </vzBrCP>
</fvTenant>
</polUni>

```

REST API Example: Tenant External Network Policy

The following XML code is an example of a Tenant Layer 3 external network policy.

```

<polUni>

  <fvTenant name='t0'>
    <fvCtx name="o1">
      <fvRsOspfCtxPol tnOspfCtxPolName="ospfCtxPol"/>
    </fvCtx>
    <fvCtx name="o2">
    </fvCtx>

    <fvBD name="bd1">
      <fvRsBDToOut tnL3extOutName='T0-o1-L3OUT-1'/>
      <fvSubnet ip='10.16.1.1/24' scope='public'/>
      <fvRsCtx tnFvCtxName="o1"/>
    </fvBD>

    <fvAp name="AP1">
      <fvAEPg name="bd1-epg1">
        <fvRsCons tnVzBrCPName="vzBrCP-1">
        </fvRsCons>
        <fvRsProv tnVzBrCPName="vzBrCP-1">
        </fvRsProv>
        <fvSubnet ip='10.16.2.1/24' scope='private'/>
        <fvSubnet ip='10.16.3.1/24' scope='private'/>
        <fvRsBd tnFvBDName="bd1"/>
        <fvRsDomAtt tDn="uni/phys-physDomP"/>
        <fvRsPathAtt
          tDn="topology/pod-1/paths-101/pathep-[eth1/40]"
          encaps='vlan-100'
          mode='regular'
          instrImedcy='immediate' />
        </fvRsPathAtt>
      </fvAEPg>

      <fvAEPg name="bd1-epg2">
        <fvRsCons tnVzBrCPName="vzBrCP-1">
        </fvRsCons>
        <fvRsProv tnVzBrCPName="vzBrCP-1">
        </fvRsProv>
        <fvSubnet ip='10.16.4.1/24' scope='private'/>
        <fvSubnet ip='10.16.5.1/24' scope='private'/>
        <fvRsBd tnFvBDName="bd1"/>
        <fvRsDomAtt tDn="uni/phys-physDomP"/>
        <fvRsPathAtt
          tDn="topology/pod-1/paths-101/pathep-[eth1/41]"
          encaps='vlan-200'
          mode='regular'
          instrImedcy='immediate' />
        </fvRsPathAtt>
      </fvAEPg>
    </fvAp>

    <l3extOut name="T0-o1-L3OUT-1">

      <l3extRsEctx tnFvCtxName="o1"/>
      <ospfExtP areaId='60'/>
      <l3extInstP name="l3extInstP-1">
        <fvRsCons tnVzBrCPName="vzBrCP-1">
        </fvRsCons>
        <fvRsProv tnVzBrCPName="vzBrCP-1">
        </fvRsProv>
        <l3extSubnet ip="192.5.1.0/24" />
        <l3extSubnet ip="192.5.2.0/24" />
        <l3extSubnet ip="192.6.0.0/16" />
        <l3extSubnet ip="199.0.0.0/8" />
      </l3extInstP>
    </l3extOut>
  </fvTenant>
</polUni>

```

```

<l3extLNodeP name="l3extLNodeP-1">
  <l3extRsNodeL3OutAtt
    tDn="topology/pod-1/node-101" rtrId="10.17.1.1">
      <ipRouteP ip="10.16.101.1/32">
        <ipNextHopP nhAddr="10.17.1.99"/>
      </ipRouteP>
      <ipRouteP ip="10.16.102.1/32">
        <ipNextHopP nhAddr="10.17.1.99"/>
      </ipRouteP>
      <ipRouteP ip="10.17.1.3/32">
        <ipNextHopP nhAddr="10.11.2.2"/>
      </ipRouteP>
    </l3extRsNodeL3OutAtt >

  <l3extLIIfP name='l3extLIIfP-1'>
    <l3extRsPathL3OutAtt
      tDn="topology/pod-1/paths-101/pathep-[eth1/25]"
      encap='vlan-1001'
      ifInstT='sub-interface'
      addr="10.11.2.1/24"
      mtu="1500"/>
    <ospfIfP>
      <ospfRsIfPol tnOspfIfPolName='ospfIfPol' />
    </ospfIfP>
  </l3extLIIfP>
</l3extLNodeP>
</l3extOut>

<ospfIfPol name="ospfIfPol" />
<ospfCtxPol name="ospfCtxPol" />

<vzFilter name="vzFilter-in-1">
  <vzEntry name="vzEntry-in-1"/>
</vzFilter>
<vzFilter name="vzFilter-out-1">
  <vzEntry name="vzEntry-out-1"/>
</vzFilter>

<vzBrCP name="vzBrCP-1">
  <vzSubj name="vzSubj-1">
    <vzInTerm>
      <vzRsFiltAtt tnVzFilterName="vzFilter-in-1"/>
    </vzInTerm>
    <vzOutTerm>
      <vzRsFiltAtt tnVzFilterName="vzFilter-out-1"/>
    </vzOutTerm>
  </vzSubj>
</vzBrCP>
</fvTenant>
</polUni>

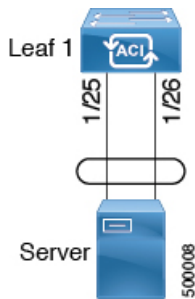
```

Layer 3 Routed and Sub-Interface Port Channels

About Layer 3 Port Channels

Previously, Cisco APIC supported only Layer 2 port channels. Starting with release 3.2(1), Cisco APIC now supports Layer 3 port channels.

Figure 35: Switch Port Channel Configuration



Note Layer 3 routed and sub-interface port channels on border leaf switches are supported only on new generation switches, which are switch models with "EX", "FX" or "FX2" at the end of the switch name.

Configuring Port Channels Using the REST API

Before you begin



Note The procedures in this section are meant specifically for configuring port channels as a prerequisite to the procedures for configuring a Layer 3 routed or sub-interface port channel. For general instructions on configuring leaf switch port channels, refer to the *Cisco APIC Basic Configuration Guide* or *Cisco APIC Layer 2 Networking Configuration Guide*.

- The ACI fabric is installed, APIC controllers are online, and the APIC cluster is formed and healthy.
- An APIC fabric administrator account is available that will enable creating the necessary fabric infrastructure configurations.
- The target leaf switches are registered in the ACI fabric and available.



Note In the following REST API example, long single lines of text are broken up with the \ character to improve readability.

To configure a port channel using the REST API, send a post with XML such as the following:

Example:

```
<polUni>
<infraInfra dn="uni/infra">
  <infraNodeP name="test1">
    <infraLeafS name="leafs" type="range">
      <infraNodeBlk name="nblk" from_"101" to_"101"/>
    </infraLeafS>
  <infraRsAccPortP tDn="uni/infra/accportprof-test1"/>
</infraNodeP>
</infraInfra>
</polUni>
```

```

</infraNodeP>
  <infraAccPortP name="test1">
    <infraHPortS name="pselc" type="range">
      <infraPortBlk name="blk1" fromCard="1" toCard="1" fromPort="18" \
        toPort="19"/>
      <infraRsAccBaseGrp tDn="uni/infra/funcprof/accbundle-pol17_PolGrp"/>
    </infraHPortS>
  </infraAccPortP>

  <infraFuncP>
    <infraAccBndlGrp name="pol17_PolGrp" lagT="link">
      <infraRsHIfPol tnFabricHIfPolName="default"/>
      <infraRsCdpIfPol tnCdpIfPolName="default"/>
      <infraRsLacpPol tnLacpLagPolName="default"/>
    </infraAccBndlGrp>
  </infraFuncP>
</infraInfra>
</polUni>

```

What to do next

Configure a Layer 3 routed port channel or sub-interface port channel using the REST API.

Configuring a Layer 3 Routed Port Channel Using the REST API

Before you begin

- The ACI fabric is installed, APIC controllers are online, and the APIC cluster is formed and healthy.
- An APIC fabric administrator account is available that will enable creating the necessary fabric infrastructure configurations.
- The target leaf switches are registered in the ACI fabric and available.
- Port channels are configured using the procedures in "Configuring Port Channels Using the REST API".



Note In the following REST API example, long single lines of text are broken up with the \ character to improve readability.

To configure a Layer 3 route to the port channels that you created previously using the REST API, send a post with XML such as the following:

Example:

```

<polUni>
  <fvTenant name=pep9>
    <l3extOut descr="" dn="uni/tn-pep9/out-routAccounting" enforceRtctrl="export" \
      name="routAccounting" nameAlias="" ownerKey="" ownerTag="" \
      targetDscp="unspecified">
      <l3extRsL3DomAtt tDn="uni/l3dom-Dom1"/>
      <l3extRsEctx tnFvCtxName="ctx9"/>
      <l3extLNodeP configIssues="" descr="" name="node101" nameAlias="" ownerKey="" \

```

```

ownerTag="" tag="yellow-green" targetDscp="unspecified">
  <l3extRsNodeL3OutAtt rtrId="10.1.0.101" rtrIdLoopBack="yes" \
    tDn="topology/pod-1/node-101">
    <l3extInfraNodeP descr="" fabricExtCtrlPeering="no" \
      fabricExtIntersiteCtrlPeering="no" name="" nameAlias="" spineRole=""/>
  </l3extRsNodeL3OutAtt>
  <l3extLIIfP descr="" name="lifp17" nameAlias="" ownerKey="" ownerTag="" \
    tag="yellow-green">
    <ospfIfP authKeyId="1" authType="none" descr="" name="" nameAlias="">
      <ospfRsIfPol tnOspfIfPolName=""/>
    </ospfIfP>
    <l3extRsPathL3OutAtt addr="10.1.5.3/24" autostate="disabled" descr="" \
      encap="unknown" encapScope="local" ifInstT="l3-port" llAddr="::" \
      mac="00:22:BD:F8:19:FF" mode="regular" mtu="inherit" \
      tDn="topology/pod-1/paths-101/pathep-[pol7_PolGrp]" \
      targetDscp="unspecified"/>
    <l3extRsNdIfPol tnNdIfPolName=""/>
    <l3extRsIngressQosDppPol tnQosDppPolName=""/>
    <l3extRsEgressQosDppPol tnQosDppPolName=""/>
  </l3extLIIfP>
</l3extLNodeP>
<l3extInstP descr="" floodOnEncap="disabled" matchT="AtleastOne" \
  name="accountingInst" nameAlias="" prefGrMemb="exclude" prio="unspecified" \
  targetDscp="unspecified">
  <fvRsProv matchT="AtleastOne" prio="unspecified" tnVzBrCPName="webCtrct"/>
  <l3extSubnet aggregate="export-rtctrl,import-rtctrl" descr="" ip="0.0.0.0/0" \
    name="" nameAlias="" scope="export-rtctrl,import-rtctrl,import-security"/>
  <l3extSubnet aggregate="export-rtctrl,import-rtctrl" descr="" ip="::/0" \
    name="" nameAlias="" scope="export-rtctrl,import-rtctrl,import-security"/>
  <fvRsCustQosPol tnQosCustomPolName=""/>
</l3extInstP>
<l3extConsLbl descr="" name="golf" nameAlias="" owner="infra" ownerKey="" \
  ownerTag="" tag="yellow-green"/>
</l3extOut>
</fvTenant>
</polUni>

```

Configuring a Layer 3 Sub-Interface Port Channel Using the REST API

Before you begin

- The ACI fabric is installed, APIC controllers are online, and the APIC cluster is formed and healthy.
- An APIC fabric administrator account is available that will enable creating the necessary fabric infrastructure configurations.
- The target leaf switches are registered in the ACI fabric and available.
- Port channels are configured using the procedures in "Configuring Port Channels Using the REST API".



Note In the following REST API example, long single lines of text are broken up with the \ character to improve readability.

To configure a Layer 3 sub-interface route to the port channels that you created previously using the REST API, send a post with XML such as the following:

Example:

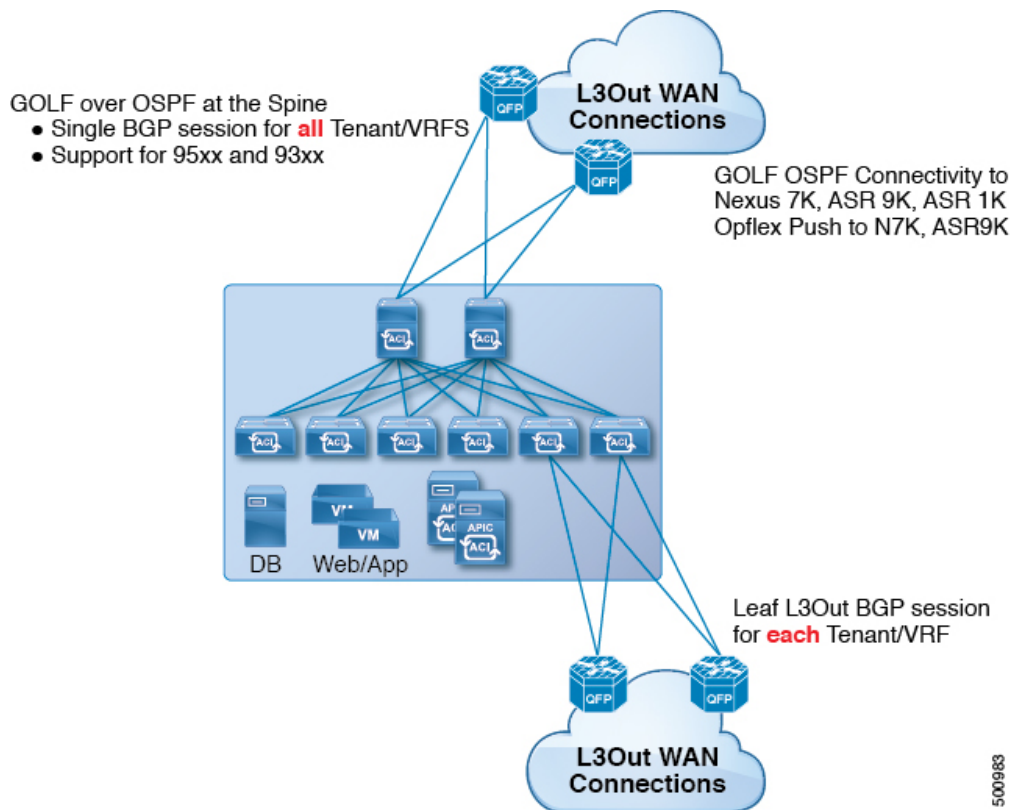
```
<polUni>
<fvTenant name=pep9>
  <l3extOut descr="" dn="uni/tn-pep9/out-routAccounting" enforceRtctrl="export" \
    name="routAccounting" nameAlias="" ownerKey="" ownerTag="" targetDscp="unspecified">
    <l3extRsL3DomAtt tDn="uni/l3dom-Dom1"/>
    <l3extRsEctx tnFvCtxName="ctx9"/>
    <l3extLNodeP configIssues="" descr="" name="node101" nameAlias="" ownerKey="" \
      ownerTag="" tag="yellow-green" targetDscp="unspecified">
      <l3extRsNodeL3OutAtt rtrId="10.1.0.101" rtrIdLoopBack="yes" \
        tDn="topology/pod-1/node-101">
        <l3extInfraNodeP descr="" fabricExtCtrlPeering="no" \
          fabricExtIntersiteCtrlPeering="no" name="" nameAlias="" spineRole=""/>
      </l3extRsNodeL3OutAtt>
      <l3extLIfP descr="" name="lifp27" nameAlias="" ownerKey="" ownerTag="" \
        tag="yellow-green">
        <ospfIfP authKeyId="1" authType="none" descr="" name="" nameAlias="">
          <ospfRsIfPol tnOspfIfPolName=""/>
        </ospfIfP>
        <l3extRsPathL3OutAtt addr="11.1.5.3/24" autostate="disabled" descr="" \
          encap="vlan-2001" encapScope="local" ifInstT="sub-interface" \
          llAddr="::" mac="00:22:BD:F8:19:FF" mode="regular" mtu="inherit" \
          tDn="topology/pod-1/paths-101/pathep-[po27_PolGrp]" \
          targetDscp="unspecified"/>
        <l3extRsNdIfPol tnNdIfPolName=""/>
        <l3extRsIngressQosDppPol tnQosDppPolName=""/>
        <l3extRsEgressQosDppPol tnQosDppPolName=""/>
      </l3extLIfP>
    </l3extLNodeP>
    <l3extInstP descr="" floodOnEncap="disabled" matchT="AtleastOne" \
      name="accountingInst" nameAlias="" prefGrMemb="exclude" prio="unspecified" \
      targetDscp="unspecified">
      <fvRsProv matchT="AtleastOne" prio="unspecified" tnVzBrCPName="webCtrct"/>
      <l3extSubnet aggregate="export-rtctrl,import-rtctrl" descr="" ip="0.0.0.0/0" \
        name="" nameAlias="" scope="export-rtctrl,import-rtctrl,import-security"/>
      <l3extSubnet aggregate="export-rtctrl,import-rtctrl" descr="" ip "::/0" \
        name="" nameAlias="" scope="export-rtctrl,import-rtctrl,import-security"/>
      <fvRsCustQosPol tnQosCustomPolName=""/>
    </l3extInstP>
    <l3extConsLbl descr="" name="golf" nameAlias="" owner="infra" ownerKey="" \
      ownerTag="" tag="yellow-green"/>
  </l3extOut>
</fvTenant>
</polUni>
```

Cisco ACI GOLF

Cisco ACI GOLF

The Cisco ACI GOLF feature (also known as Layer 3 EVPN Services for Fabric WAN) enables much more efficient and scalable ACI fabric WAN connectivity. It uses the BGP EVPN protocol over OSPF for WAN routers that are connected to spine switches.

Figure 36: Cisco ACI GOLF Topology



All tenant WAN connections use a single session on the spine switches where the WAN routers are connected. This aggregation of tenant BGP sessions towards the Data Center Interconnect Gateway (DCIG) improves control plane scale by reducing the number of tenant BGP sessions and the amount of configuration required for all of them. The network is extended out using Layer 3 subinterfaces configured on spine fabric ports. Transit routing with shared services using GOLF is not supported.

A Layer 3 external outside network (`L3extOut`) for GOLF physical connectivity for a spine switch is specified under the `infra` tenant, and includes the following:

- `LNodeP` (`L3extInstP` is not required within the `L3Out` in the `infra` tenant.)
- A provider label for the `L3extOut` for GOLF in the `infra` tenant.
- OSPF protocol policies

- BGP protocol policies

All regular tenants use the above-defined physical connectivity. The `L3extOut` defined in regular tenants requires the following:

- An `L3extInstP` (EPG) with subnets and contracts. The scope of the subnet is used to control import/export route control and security policies. The bridge domain subnet must be set to advertise externally and it must be in the same VRF as the application EPG and the GOLF `L3Out` EPG.
- Communication between the application EPG and the GOLF `L3Out` EPG is governed by explicit contracts (not Contract Preferred Groups).
- An `L3extConsLbl` consumer label that must be matched with the same provider label of an `L3Out` for GOLF in the `infra` tenant. Label matching enables application EPGs in other tenants to consume the `LNodeP` external `L3Out` EPG.
- The BGP EVPN session in the matching provider `L3extOut` in the `infra` tenant advertises the tenant routes defined in this `L3Out`.

Configuring GOLF Using the REST API

SUMMARY STEPS

1. The following example shows how to deploy nodes and spine switch interfaces for GOLF, using the REST API:
2. The XML below configures the spine switch interfaces and infra tenant provider of the GOLF service. Include this XML structure in the body of the POST message.
3. The XML below configures the tenant consumer of the infra part of the GOLF service. Include this XML structure in the body of the POST message.

DETAILED STEPS

Step 1 The following example shows how to deploy nodes and spine switch interfaces for GOLF, using the REST API:

Example:

```
POST
https://192.0.20.123/api/mo/uni/golf.xml
```

Step 2 The XML below configures the spine switch interfaces and infra tenant provider of the GOLF service. Include this XML structure in the body of the POST message.

Example:

```
<l3extOut descr="" dn="uni/tn-infra/out-golf" enforceRtctrl="export,import"
  name="golf"
  ownerKey="" ownerTag="" targetDscp="unspecified">
  <l3extRsEctx tnFvCtxName="overlay-1"/>
  <l3extProvLbl descr="" name="golf"
    ownerKey="" ownerTag="" tag="yellow-green"/>
  <l3extLNodeP configIssues="" descr=""
    name="bLeaf" ownerKey="" ownerTag=""
    tag="yellow-green" targetDscp="unspecified">
  <l3extRsNodeL3OutAtt rtrId="10.10.3.3" rtrIdLoopBack="no"
    tDn="topology/pod-1/node-111">
```

```

    <l3extInfraNodeP descr="" fabricExtCtrlPeering="yes" name=""/>
    <l3extLoopBackIfP addr="10.10.3.3" descr="" name=""/>
  </l3extRsNodeL3OutAtt>
  <l3extRsNodeL3OutAtt rtrId="10.10.3.4" rtrIdLoopBack="no"
    tDn="topology/pod-1/node-112">
    <l3extInfraNodeP descr="" fabricExtCtrlPeering="yes" name=""/>
    <l3extLoopBackIfP addr="10.10.3.4" descr="" name=""/>
  </l3extRsNodeL3OutAtt>
  <l3extLIfP descr="" name="portIf-spine1-3"
    ownerKey="" ownerTag="" tag="yellow-green">
    <ospfIfP authKeyId="1" authType="none" descr="" name="">
      <ospfRsIfPol tnOspfIfPolName="ospfIfPol"/>
    </ospfIfP>
    <l3extRsNdIfPol tnNdIfPolName=""/>
    <l3extRsIngressQosDppPol tnQosDppPolName=""/>
    <l3extRsEgressQosDppPol tnQosDppPolName=""/>
    <l3extRsPathL3OutAtt addr="7.2.1.1/24" descr=""
      encap="vlan-4"
      encapScope="local"
      ifInstT="sub-interface"
      llAddr="::" mac="00:22:BD:F8:19:FF"
      mode="regular"
      mtu="1500"
      tDn="topology/pod-1/paths-111/pathep-[eth1/12]"
      targetDscp="unspecified"/>
  </l3extLIfP>
  <l3extLIfP descr="" name="portIf-spine2-1"
    ownerKey=""
    ownerTag=""
    tag="yellow-green">
    <ospfIfP authKeyId="1"
      authType="none"
      descr=""
      name="">
      <ospfRsIfPol tnOspfIfPolName="ospfIfPol"/>
    </ospfIfP>
    <l3extRsNdIfPol tnNdIfPolName=""/>
    <l3extRsIngressQosDppPol tnQosDppPolName=""/>
    <l3extRsEgressQosDppPol tnQosDppPolName=""/>
    <l3extRsPathL3OutAtt addr="7.1.0.1/24" descr=""
      encap="vlan-4"
      encapScope="local"
      ifInstT="sub-interface"
      llAddr="::" mac="00:22:BD:F8:19:FF"
      mode="regular"
      mtu="9000"
      tDn="topology/pod-1/paths-112/pathep-[eth1/11]"
      targetDscp="unspecified"/>
  </l3extLIfP>
  <l3extLIfP descr="" name="portif-spine2-2"
    ownerKey=""
    ownerTag=""
    tag="yellow-green">
    <ospfIfP authKeyId="1"
      authType="none" descr=""
      name="">
      <ospfRsIfPol tnOspfIfPolName="ospfIfPol"/>
    </ospfIfP>
    <l3extRsNdIfPol tnNdIfPolName=""/>
    <l3extRsIngressQosDppPol tnQosDppPolName=""/>
    <l3extRsEgressQosDppPol tnQosDppPolName=""/>
    <l3extRsPathL3OutAtt addr="7.2.2.1/24" descr=""
      encap="vlan-4"
      encapScope="local"

```

```

        ifInstT="sub-interface"
            llAddr="::" mac="00:22:BD:F8:19:FF"
            mode="regular"
            mtu="1500"
            tDn="topology/pod-1/paths-112/pathep-[eth1/12]"
            targetDscp="unspecified"/>
    </l3extLIIfP>
    <l3extLIIfP descr="" name="portIf-spine1-2"
        ownerKey="" ownerTag="" tag="yellow-green">
        <ospfIfP authKeyId="1" authType="none" descr="" name="">
            <ospfRsIfPol tnOspfIfPolName="ospfIfPol"/>
        </ospfIfP>
        <l3extRsNdIfPol tnNdIfPolName=""/>
        <l3extRsIngressQosDppPol tnQosDppPolName=""/>
        <l3extRsEgressQosDppPol tnQosDppPolName=""/>
        <l3extRsPathL3OutAtt addr="9.0.0.1/24" descr=""
            encap="vlan-4"
            encapScope="local"
            ifInstT="sub-interface"
                llAddr="::" mac="00:22:BD:F8:19:FF"
                mode="regular"
                mtu="9000"
                tDn="topology/pod-1/paths-111/pathep-[eth1/11]"
                targetDscp="unspecified"/>
        </l3extLIIfP>
    <l3extLIIfP descr="" name="portIf-spine1-1"
        ownerKey="" ownerTag="" tag="yellow-green">
        <ospfIfP authKeyId="1" authType="none" descr="" name="">
            <ospfRsIfPol tnOspfIfPolName="ospfIfPol"/>
        </ospfIfP>
        <l3extRsNdIfPol tnNdIfPolName=""/>
        <l3extRsIngressQosDppPol tnQosDppPolName=""/>
        <l3extRsEgressQosDppPol tnQosDppPolName=""/>
        <l3extRsPathL3OutAtt addr="7.0.0.1/24" descr=""
            encap="vlan-4"
            encapScope="local"
            ifInstT="sub-interface"
                llAddr="::" mac="00:22:BD:F8:19:FF"
                mode="regular"
                mtu="1500"
                tDn="topology/pod-1/paths-111/pathep-[eth1/10]"
                targetDscp="unspecified"/>
        </l3extLIIfP>
    <bgpInfraPeerP addr="10.10.3.2"
        allowedSelfAsCnt="3"
        ctrl="send-com,send-ext-com"
        descr="" name="" peerCtrl=""
        peerT="wan"
        privateASctrl="" ttl="2" weight="0">
        <bgpRsPeerPfxPol tnBgpPeerPfxPolName=""/>
        <bgpAsP asn="150" descr="" name="aspn"/>
    </bgpInfraPeerP>
    <bgpInfraPeerP addr="10.10.4.1"
        allowedSelfAsCnt="3"
        ctrl="send-com,send-ext-com" descr="" name="" peerCtrl=""
        peerT="wan"
        privateASctrl="" ttl="1" weight="0">
        <bgpRsPeerPfxPol tnBgpPeerPfxPolName=""/>
        <bgpAsP asn="100" descr="" name=""/>
    </bgpInfraPeerP>
    <bgpInfraPeerP addr="10.10.3.1"
        allowedSelfAsCnt="3"
        ctrl="send-com,send-ext-com" descr="" name="" peerCtrl=""
        peerT="wan"

```

```

        privateASctrl="" ttl="1" weight="0">
        <bgpRsPeerPfxPol tnBgpPeerPfxPolName=""/>
        <bgpAsP asn="100" descr="" name=""/>
    </bgpInfraPeerP>
</l3extLNodeP>
<bgpRtTargetInstrP descr="" name="" ownerKey="" ownerTag="" rtTargetT="explicit"/>
<l3extRsL3DomAtt tDn="uni/l3dom-l3dom"/>
<l3extInstP descr="" matchT="AtleastOne" name="golfInstP"
    prio="unspecified"
    targetDscp="unspecified">
    <fvRsCustQosPol tnQosCustomPolName=""/>
</l3extInstP>
<bgpExtP descr=""/>
<ospfExtP areaCost="1"
    areaCtrl="redistribute,summary"
    areaId="0.0.0.1"
    areaType="regular" descr=""/>
</l3extOut>

```

Step 3 The XML below configures the tenant consumer of the infra part of the GOLF service. Include this XML structure in the body of the POST message.

Example:

```

<fvTenant descr="" dn="uni/tn-pep6" name="pep6" ownerKey="" ownerTag="">
  <vzBrCP descr="" name="webCtrct"
    ownerKey="" ownerTag="" prio="unspecified"
    scope="global" targetDscp="unspecified">
    <vzSubj consMatchT="AtleastOne" descr=""
      name="http" prio="unspecified" provMatchT="AtleastOne"
      revFltPorts="yes" targetDscp="unspecified">
      <vzRsSubjFiltAtt directives="" tnVzFilterName="default"/>
    </vzSubj>
  </vzBrCP>
  <vzBrCP descr="" name="webCtrct-pod2"
    ownerKey="" ownerTag="" prio="unspecified"
    scope="global" targetDscp="unspecified">
    <vzSubj consMatchT="AtleastOne" descr=""
      name="http" prio="unspecified"
      provMatchT="AtleastOne" revFltPorts="yes"
      targetDscp="unspecified">
      <vzRsSubjFiltAtt directives=""
        tnVzFilterName="default"/>
    </vzSubj>
  </vzBrCP>
  <fvCtx descr="" knwMcastAct="permit"
    name="ctx6" ownerKey="" ownerTag=""
    pcEnfDir="ingress" pcEnfPref="enforced">
    <bgpRtTargetP af="ipv6-ucast"
      descr="" name="" ownerKey="" ownerTag="">
      <bgpRtTarget descr="" name="" ownerKey="" ownerTag=""
        rt="route-target:as4-nn2:100:1256"
        type="export"/>
      <bgpRtTarget descr="" name="" ownerKey="" ownerTag=""
        rt="route-target:as4-nn2:100:1256"
        type="import"/>
    </bgpRtTargetP>
    <bgpRtTargetP af="ipv4-ucast"
      descr="" name="" ownerKey="" ownerTag="">
      <bgpRtTarget descr="" name="" ownerKey="" ownerTag=""
        rt="route-target:as4-nn2:100:1256"
        type="export"/>
      <bgpRtTarget descr="" name="" ownerKey="" ownerTag=""
        rt="route-target:as4-nn2:100:1256"

```

```

        type="import"/>
    </bgpRtTargetP>
    <fvRsCtxToExtRouteTagPol tnL3extRouteTagPolName=""/>
    <fvRsBgpCtxPol tnBgpCtxPolName=""/>
    <vzAny descr="" matchT="AtleastOne" name=""/>
    <fvRsOspfCtxPol tnOspfCtxPolName=""/>
    <fvRsCtxToEpRet tnFvEpRetPolName=""/>
    <l3extGlobalCtxName descr="" name="dci-pep6"/>
</fvCtx>
<fvBD arpFlood="no" descr="" epMoveDetectMode=""
  ipLearning="yes"
  limitIpLearnToSubnets="no"
  llAddr="::" mac="00:22:BD:F8:19:FF"
  mcastAllow="no"
  multiDstPktAct="bd-flood"
  name="bd107" ownerKey="" ownerTag="" type="regular"
  unicastRoute="yes"
  unkMacUcastAct="proxy"
  unkMcastAct="flood"
  vmac="not-applicable">
  <fvRsBDToNdP tnNdIfPolName=""/>
  <fvRsBDToOut tnL3extOutName="routAccounting-pod2"/>
  <fvRsCtx tnFvCtxName="ctx6"/>
  <fvRsIgmpsn tnIgmpSnoopPolName=""/>
  <fvSubnet ctrl="" descr="" ip="27.6.1.1/24"
    name="" preferred="no"
    scope="public"
    virtual="no"/>
  <fvSubnet ctrl="nd" descr="" ip="2001:27:6:1::1/64"
    name="" preferred="no"
    scope="public"
    virtual="no">
    <fvRsNdPfxPol tnNdPfxPolName=""/>
  </fvSubnet>
  <fvRsBdToEpRet resolveAct="resolve" tnFvEpRetPolName=""/>
</fvBD>
<fvBD arpFlood="no" descr="" epMoveDetectMode=""
  ipLearning="yes"
  limitIpLearnToSubnets="no"
  llAddr="::" mac="00:22:BD:F8:19:FF"
  mcastAllow="no"
  multiDstPktAct="bd-flood"
  name="bd103" ownerKey="" ownerTag="" type="regular"
  unicastRoute="yes"
  unkMacUcastAct="proxy"
  unkMcastAct="flood"
  vmac="not-applicable">
  <fvRsBDToNdP tnNdIfPolName=""/>
  <fvRsBDToOut tnL3extOutName="routAccounting"/>
  <fvRsCtx tnFvCtxName="ctx6"/>
  <fvRsIgmpsn tnIgmpSnoopPolName=""/>
  <fvSubnet ctrl="" descr="" ip="23.6.1.1/24"
    name="" preferred="no"
    scope="public"
    virtual="no"/>
  <fvSubnet ctrl="nd" descr="" ip="2001:23:6:1::1/64"
    name="" preferred="no"
    scope="public" virtual="no">
    <fvRsNdPfxPol tnNdPfxPolName=""/>
  </fvSubnet>
  <fvRsBdToEpRet resolveAct="resolve" tnFvEpRetPolName=""/>
</fvBD>
<vnsSvcCont/>
<fvRsTenantMonPol tnMonEPGPName=""/>

```

```

<fvAp descr="" name="AP1"
  ownerKey="" ownerTag="" prio="unspecified">
  <fvAEPg descr=""
    isAttrBasedEPg="no"
    matchT="AtleastOne"
    name="epg107"
    pcEnfPref="unenforced" prio="unspecified">
    <fvRsCons prio="unspecified"
      tnVzBrCPName="webCtrct-pod2"/>
    <fvRsPathAtt descr=""
      encap="vlan-1256"
      instrImedcy="immediate"
      mode="regular" primaryEncap="unknown"
      tDn="topology/pod-2/paths-107/pathep-[eth1/48]"/>
    <fvRsDomAtt classPref="encap" delimiter=""
      encap="unknown"
      instrImedcy="immediate"
      primaryEncap="unknown"
      resImedcy="lazy" tDn="uni/phys-phys"/>
    <fvRsCustQosPol tnQosCustomPolName=""/>
    <fvRsBd tnFvBDName="bd107"/>
    <fvRsProv matchT="AtleastOne"
      prio="unspecified"
      tnVzBrCPName="default"/>
  </fvAEPg>
  <fvAEPg descr=""
    isAttrBasedEPg="no"
    matchT="AtleastOne"
    name="epg103"
    pcEnfPref="unenforced" prio="unspecified">
    <fvRsCons prio="unspecified" tnVzBrCPName="default"/>
    <fvRsCons prio="unspecified" tnVzBrCPName="webCtrct"/>
    <fvRsPathAtt descr="" encap="vlan-1256"
      instrImedcy="immediate"
      mode="regular" primaryEncap="unknown"
      tDn="topology/pod-1/paths-103/pathep-[eth1/48]"/>
    <fvRsDomAtt classPref="encap" delimiter=""
      encap="unknown"
      instrImedcy="immediate"
      primaryEncap="unknown"
      resImedcy="lazy" tDn="uni/phys-phys"/>
    <fvRsCustQosPol tnQosCustomPolName=""/>
    <fvRsBd tnFvBDName="bd103"/>
  </fvAEPg>
</fvAp>
<l3extOut descr=""
  enforceRtctrl="export"
  name="routAccounting-pod2"
  ownerKey="" ownerTag="" targetDscp="unspecified">
  <l3extRsEctx tnFvCtxName="ctx6"/>
  <l3extInstP descr=""
    matchT="AtleastOne"
    name="accountingInst-pod2"
    prio="unspecified" targetDscp="unspecified">
  <l3extSubnet aggregate="export-rtctrl,import-rtctrl"
    descr="" ip="::/0" name=""
    scope="export-rtctrl,import-rtctrl,import-security"/>
  <l3extSubnet aggregate="export-rtctrl,import-rtctrl"
    descr=""
    ip="0.0.0.0/0" name=""
    scope="export-rtctrl,import-rtctrl,import-security"/>
  <fvRsCustQosPol tnQosCustomPolName=""/>
  <fvRsProv matchT="AtleastOne"
    prio="unspecified" tnVzBrCPName="webCtrct-pod2"/>

```



```

</l3extInstP>
<l3extConsLbl descr=""
  name="golf2"
  owner="infra"
  ownerKey="" ownerTag="" tag="yellow-green"/>
</l3extOut>
<l3extOut descr=""
  enforceRtctrl="export"
  name="routAccounting"
  ownerKey="" ownerTag="" targetDscp="unspecified">
<l3extRsEctx tnFvCtxName="ctx6"/>
<l3extInstP descr=""
  matchT="AtleastOne"
  name="accountingInst"
  prio="unspecified" targetDscp="unspecified">
<l3extSubnet aggregate="export-rtctrl,import-rtctrl" descr=""
  ip="0.0.0.0/0" name=""
  scope="export-rtctrl,import-rtctrl,import-security"/>
<fvRsCustQosPol tnQosCustomPolName=""/>
<fvRsProv matchT="AtleastOne" prio="unspecified" tnVzBrCPName="webCtrct"/>
</l3extInstP>
<l3extConsLbl descr=""
  name="golf"
  owner="infra"
  ownerKey="" ownerTag="" tag="yellow-green"/>
</l3extOut>
</fvTenant>

```

Distributing BGP EVPN Type-2 Host Routes to a DCIG

In APIC up to release 2.0(1f), the fabric control plane did not send EVPN host routes directly, but advertised public bridge domain (BD) subnets in the form of BGP EVPN type-5 (IP Prefix) routes to a Data Center Interconnect Gateway (DCIG). This could result in suboptimal traffic forwarding. To improve forwarding, in APIC release 2.1x, you can enable fabric spines to also advertise host routes using EVPN type-2 (MAC-IP) host routes to the DCIG along with the public BD subnets.

To do so, you must perform the following steps:

1. When you configure the BGP Address Family Context Policy, enable Host Route Leak.
2. When you leak the host route to BGP EVPN in a GOLF setup:
 - a. To enable host routes when GOLF is enabled, the BGP Address Family Context Policy must be configured under the application tenant (the application tenant is the consumer tenant that leaks the endpoint to BGP EVPN) rather than under the infrastructure tenant.
 - b. For a single-pod fabric, the host route feature is not required. The host route feature is required to avoid sub-optimal forwarding in a multi-pod fabric setup. However, if a single-pod fabric is setup, then in order to leak the endpoint to BGP EVPN, a Fabric External Connection Policy must be configured to provide the ETEP IP address. Otherwise, the host route will not leak to BGP EVPN.
3. When you configure VRF properties:
 - a. Add the BGP Address Family Context Policy to the BGP Context Per Address Families for IPv4 and IPv6.

- b. Configure BGP Route Target Profiles that identify routes that can be imported or exported from the VRF.

Enabling Distributing BGP EVPN Type-2 Host Routes to a DCIG Using the REST API

Enable distributing BGP EVPN type-2 host routes using the REST API, as follows:

Before you begin

EVPN services must be configured.

Step 1 Configure the Host Route Leak policy, with a POST containing XML such as in the following example:

Example:

```
<bgpCtxAfPol descr="" ctrl="host-rt-leak" name="bgpCtxPol_0 status=""/>
```

Step 2 Apply the policy to the VRF BGP Address Family Context Policy for one or both of the address families using a POST containing XML such as in the following example:

Example:

```
<fvCtx name="vni-10001">
<fvRsCtxToBgpCtxAfPol af="ipv4-ucast" tnBgpCtxAfPolName="bgpCtxPol_0"/>
<fvRsCtxToBgpCtxAfPol af="ipv6-ucast" tnBgpCtxAfPolName="bgpCtxPol_0"/>
</fvCtx>
```

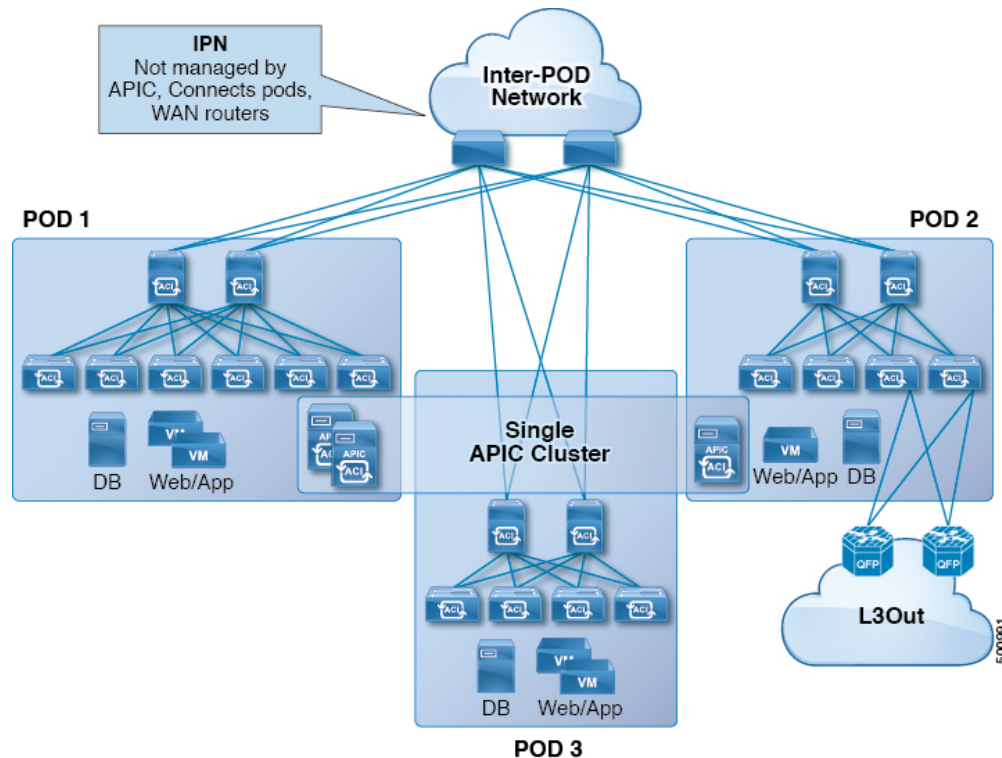
Multipod

Multipod

Multipod enables provisioning a more fault tolerant fabric comprised of multiple pods with isolated control plane protocols. Also, multipod provides more flexibility with regard to the full mesh cabling between leaf and spine switches. For example, if leaf switches are spread across different floors or different buildings, multipod enables provisioning multiple pods per floor or building and providing connectivity between pods through spine switches.

Multipod uses MP-BGP EVPN as the control-plane communication protocol between the ACI spines in different Pods. WAN routers can be provisioned in the IPN, directly connected to spine switches, or connected to border leaf switches. Multipod uses a single APIC cluster for all the pods; all the pods act as a single fabric. Individual APIC controllers are placed across the pods but they are all part of a single APIC cluster.

Figure 37: Multipod Overview



For control plane isolation, IS-IS and COOP are not extended across pods. Endpoints synchronize across pods using BGP EVPN over the IPN between the pods. Two spines in each pod are configured to have BGP EVPN sessions with spines of other pods. The spines connected to the IPN get the endpoints and multicast groups from COOP within a pod, but they advertise them over the IPN EVPN sessions between the pods. On the receiving side, BGP gives them back to COOP and COOP synchs them across all the spines in the pod. WAN routes are exchanged between the pods using BGP VPNv4/VPNv6 address families; they are not exchanged using the EVPN address family.

There are two modes of setting up the spine switches for communicating across pods as peers and route reflectors:

- **Automatic**

- Automatic mode is a route reflector based mode that does not support a full mesh where all spines peer with each other. The administrator must post an existing BGP route reflector policy and select IPN aware (EVPN) route reflectors. All the peer/client settings are automated by the APIC.
- The administrator does not have an option to choose route reflectors that don't belong to the fabric (for example, in the IPN).

- **Manual**

- The administrator has the option to configure full mesh where all spines peer with each other without route reflectors.
- In manual mode, the administrator must post the already existing BGP peer policy.

Observe the following multipod guidelines and limitations:

- When adding a pod to the ACI fabric, wait for the control plane to converge before adding another pod.
- OSPF is deployed on ACI spine switches and IPN switches to provide reachability between PODs. Layer 3 subinterfaces are created on spines to connect to IPN switches. OSPF is enabled on these Layer 3 subinterfaces and per POD TEP prefixes are advertised over OSPF. There is one subinterface created on each external spine link. Provision many external links on each spine if the expectation is that the amount of east-west traffic between PODs will be large. Currently, ACI spine switches support up to 64 external links on each spine, and each subinterface can be configured for OSPF. Spine proxy TEP addresses are advertised in OSPF over all the subinterfaces leading to a maximum of 64 way ECMP on the IPN switch for proxy TEP addresses. Similarly, spines would receive proxy TEP addresses of other PODs from IPN switches over OSPF and the spine can have up to 64 way ECMP for remote pod proxy TEP addresses. In this way, traffic between PODs spread over all these external links provides the desired bandwidth.
- When the all fabric links of a spine switch are down, OSPF advertises the TEP routes with the maximum metric. This will force the IPN switch to remove the spine switch from ECMP which will prevent the IPN from forwarding traffic to the down spine switch. Traffic is then received by other spines that have up fabric links.
- Up to APIC release 2.0(2), multipod is not supported with GOLF. In release 2.0 (2) the two features are supported in the same fabric only over Cisco Nexus N9000K switches without “EX” on the end of the switch name; for example, N9K-9312TX. Since the 2.1(1) release, the two features can be deployed together over all the switches used in the multipod and EVPN topologies.
- In a multipod fabric, if a spine in POD1 uses the infra tenant L3extOut-1, the TORs for the other pods (POD2, POD3) cannot use the same infra L3extOut (L3extOut-1) for Layer 3 EVPN control plane connectivity. Each POD must use their own spine switch and infra L3extOut, because it is not supported to use a pod as a transit for WAN connectivity of other pods.
- No filtering is done for limiting the routes exchanged across pods. All end-point and WAN routes present in each pod are exported to other pods.
- Inband management across pods is automatically configured by a self tunnel on every spine.
- The maximum latency supported between pods is 10 msec RTT, which roughly translates to a geographical distance of up to 500 miles.

Setting Up Multi-Pod Fabric Using the REST API

Step 1 Login to Cisco APIC:

Example:

```
http://<apic-name/ip>:80/api/aaaLogin.xml
data: <aaaUser name="admin" pwd="ins3965!" />
```

Step 2 Configure the TEP pool:

Example:

```
http://<apic-name/ip>:80/api/policymgr/mo/uni/controller.xml
<fabricSetupPol status=''>
  <fabricSetupP podId="1" tepPool="10.0.0.0/16" />
```

```
<fabricSetupP podId="2" tepPool="10.1.0.0/16" status='' />
</fabricSetupPol>
```

Step 3 Configure the node ID policy:

Example:

<http://<apic-name/ip>:80/api/node/mo/uni/controller.xml>

```
<fabricNodeIdentPol>
<fabricNodeIdentP serial="SAL1819RXP4" name="ifav4-leaf1" nodeId="101" podId="1"/>
<fabricNodeIdentP serial="SAL1803L25H" name="ifav4-leaf2" nodeId="102" podId="1"/>
<fabricNodeIdentP serial="SAL1934MNY0" name="ifav4-leaf3" nodeId="103" podId="1"/>
<fabricNodeIdentP serial="SAL1934MNY3" name="ifav4-leaf4" nodeId="104" podId="1"/>
<fabricNodeIdentP serial="SAL1748H56D" name="ifav4-spine1" nodeId="201" podId="1"/>
<fabricNodeIdentP serial="SAL1938P7A6" name="ifav4-spine3" nodeId="202" podId="1"/>
<fabricNodeIdentP serial="SAL1938PHBB" name="ifav4-leaf5" nodeId="105" podId="2"/>
<fabricNodeIdentP serial="SAL1942R857" name="ifav4-leaf6" nodeId="106" podId="2"/>
<fabricNodeIdentP serial="SAL1931LA3B" name="ifav4-spine2" nodeId="203" podId="2"/>
<fabricNodeIdentP serial="FGE173400A9" name="ifav4-spine4" nodeId="204" podId="2"/>
</fabricNodeIdentPol>
```

Step 4 Configure infra L3Out and external connectivity profile:

Example:

<http://<apic-name/ip>:80/api/node/mo/uni.xml>

```
<polUni>
<fvTenant descr="" dn="uni/tn-infra" name="infra" ownerKey="" ownerTag="">
  <l3extOut descr="" enforceRtctrl="export" name="multipod" ownerKey="" ownerTag=""
targetDscp="unspecified" status=''>
  <ospfExtP areaId='0' areaType='regular' status=''/>
  <l3extRsEctx tnFvCtxName="overlay-1"/>
  <l3extProvLbl descr="" name="prov_mp1" ownerKey="" ownerTag="" tag="yellow-green"/>
  <l3extLNodeP name="bSpine">
    <l3extRsNodeL3OutAtt rtrId="201.201.201.201" rtrIdLoopBack="no" tDn="topology/pod-1/node-201">
      <l3extInfraNodeP descr="" fabricExtCtrlPeering="yes" name=""/>
      <l3extLoopBackIfP addr="201::201/128" descr="" name=""/>
      <l3extLoopBackIfP addr="201.201.201.201/32" descr="" name=""/>
    </l3extRsNodeL3OutAtt>
    <l3extRsNodeL3OutAtt rtrId="202.202.202.202" rtrIdLoopBack="no" tDn="topology/pod-1/node-202">
      <l3extInfraNodeP descr="" fabricExtCtrlPeering="yes" name=""/>
      <l3extLoopBackIfP addr="202::202/128" descr="" name=""/>
      <l3extLoopBackIfP addr="202.202.202.202/32" descr="" name=""/>
    </l3extRsNodeL3OutAtt>
    <l3extRsNodeL3OutAtt rtrId="203.203.203.203" rtrIdLoopBack="no" tDn="topology/pod-2/node-203">
      <l3extInfraNodeP descr="" fabricExtCtrlPeering="yes" name=""/>
      <l3extLoopBackIfP addr="203::203/128" descr="" name=""/>
      <l3extLoopBackIfP addr="203.203.203.203/32" descr="" name=""/>
    </l3extRsNodeL3OutAtt>
    <l3extRsNodeL3OutAtt rtrId="204.204.204.204" rtrIdLoopBack="no" tDn="topology/pod-2/node-204">
      <l3extInfraNodeP descr="" fabricExtCtrlPeering="yes" name=""/>
      <l3extLoopBackIfP addr="204::204/128" descr="" name=""/>
      <l3extLoopBackIfP addr="204.204.204.204/32" descr="" name=""/>
    </l3extRsNodeL3OutAtt>
  </l3extLNodeP>
</l3extOut>
</fvTenant>
```

```

    </l3extRsNodeL3OutAtt>

    <l3extLIIfP name='portIf'>
      <l3extRsPathL3OutAtt descr='asr' tDn="topology/pod-1/paths-201/pathep-[eth1/1]"
    encap='vlan-4' ifInstT='sub-interface' addr="201.1.1.1/30" />
      <l3extRsPathL3OutAtt descr='asr' tDn="topology/pod-1/paths-201/pathep-[eth1/2]"
    encap='vlan-4' ifInstT='sub-interface' addr="201.2.1.1/30" />
      <l3extRsPathL3OutAtt descr='asr' tDn="topology/pod-1/paths-202/pathep-[eth1/2]"
    encap='vlan-4' ifInstT='sub-interface' addr="202.1.1.1/30" />
      <l3extRsPathL3OutAtt descr='asr' tDn="topology/pod-2/paths-203/pathep-[eth1/1]"
    encap='vlan-4' ifInstT='sub-interface' addr="203.1.1.1/30" />
      <l3extRsPathL3OutAtt descr='asr' tDn="topology/pod-2/paths-203/pathep-[eth1/2]"
    encap='vlan-4' ifInstT='sub-interface' addr="203.2.1.1/30" />
      <l3extRsPathL3OutAtt descr='asr' tDn="topology/pod-2/paths-204/pathep-[eth4/31]"
    encap='vlan-4' ifInstT='sub-interface' addr="204.1.1.1/30" />

      <ospfIfP>
        <ospfRsIfPol tnOspfIfPolName='ospfIfPol' />
      </ospfIfP>

    </l3extLIIfP>
  </l3extLNodeP>

  <l3extInstP descr="" matchT="AtleastOne" name="instpl" prio="unspecified"
targetDscp="unspecified">
    <fvRsCustQosPol tnQosCustomPolName="" />
  </l3extInstP>
</l3extOut>

<fvFabricExtConnP descr="" id="1" name="Fabric_Ext_Conn_Poll" rt="extended:as2-nn4:5:16" status=''>

  <fvPodConnP descr="" id="1" name="">
    <fvIp addr="100.11.1.1/32" />
  </fvPodConnP>
  <fvPodConnP descr="" id="2" name="">
    <fvIp addr="200.11.1.1/32" />
  </fvPodConnP>
  <fvPeeringP descr="" name="" ownerKey="" ownerTag="" type="automatic_with_full_mesh" />
  <l3extFabricExtRoutingP descr="" name="ext_routing_prof_1" ownerKey="" ownerTag="">
    <l3extSubnet aggregate="" descr="" ip="100.0.0.0/8" name="" scope="import-security" />
    <l3extSubnet aggregate="" descr="" ip="200.0.0.0/8" name="" scope="import-security" />
    <l3extSubnet aggregate="" descr="" ip="201.1.0.0/16" name="" scope="import-security" />
    <l3extSubnet aggregate="" descr="" ip="201.2.0.0/16" name="" scope="import-security" />
    <l3extSubnet aggregate="" descr="" ip="202.1.0.0/16" name="" scope="import-security" />
    <l3extSubnet aggregate="" descr="" ip="203.1.0.0/16" name="" scope="import-security" />
    <l3extSubnet aggregate="" descr="" ip="203.2.0.0/16" name="" scope="import-security" />
    <l3extSubnet aggregate="" descr="" ip="204.1.0.0/16" name="" scope="import-security" />
  </l3extFabricExtRoutingP>
</fvFabricExtConnP>
</fvTenant>
</polUni>

```

Anycast Services

About Anycast Services

Anycast services are supported in the Cisco ACI fabric. A typical use case is to support Cisco Adaptive Security Appliance (ASA) firewalls in the pods of a multipod fabric, but Anycast could be used to enable other services, such as DNS servers or printing services. In the ASA use case, a firewall is installed in every pod and Anycast is enabled, so the firewall can be offered as an Anycast service. One instance of a firewall going down does not affect clients, as the requests are routed to the next, nearest instance available. You install ASA firewalls in each pod, then enable Anycast and configure the IP address and MAC addresses to be used.

APIC deploys the configuration of the Anycast MAC and IP addresses to the leaf switches where the VRF is deployed or where there is a contract to allow an Anycast EPG.

Initially, each leaf switch installs the Anycast MAC and IP addresses as a proxy route to the spine switch. When the first packet from the Anycast Service is received, the destination information for the service is installed on the leaf switch behind which the service is installed. All other leaf switches continue to point to the spine proxy. When the Anycast service has been learned, located behind a leaf in a pod, COOP installs the entry on the spine switch to point to the service that is local to the pod.

When the Anycast service is running in one pod, the spine receives the route information for the Anycast service present in the pod through BGP-EVPN. If the Anycast service is already locally present, then COOP caches the Anycast service information of the remote pod. This route through the remote pod is only installed when the local instance of the service goes down.

Configuring Anycast Services Using the REST API

These examples show how to configure Anycast services in three methods:

- Behind an EPG.
- As part of a Layer 4 to Layer 7 Service Graph with Policy Based Redirect (PBR)
- As part of a Layer 4 to Layer 7 Service Graph without PBR

Before you begin

- The tenant, application profile, and application EPG have been created.
- The node group and L3Out policies have already been created.
- The Interpod Network (IPN) is already configured.
- Multipod is configured.
- In each pod, the spine switch used to connect to the IPN is also connected to at least one leaf switch.
- ASA firewalls are installed in each pod.

Step 1 To configure Anycast services behind an EPG, send a post with XML such as the following example:

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- /api/policymgr/mo/.xml -->
<polUni>
  <fvTenant name="tn1" status="created,modified">
    <fvAp name="a0">
      <fvAEPg name="web">
        <fvSubnet ctrl="no-default-gateway" ip="200.50.3.4/32" scope="private">
          <fvEpAnycast mac="00:44:55:66:55:01"/>
        </fvSubnet>
        <fvRsDomAtt tDn="uni/phys-test"/>
        <fvRsBd tnFvBDName="lab"/>
      </fvAEPg>
    </fvAp>
  </fvTenant>
</polUni>
```

Step 2 To configure Anycast services as part of a Layer 4 to Layer 7 service graph with PBR, send a post with XML such as the following example:

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- /api/policymgr/mo/.xml -->
<polUni>
  <fvTenant name="tn1" >
    <vnsSvcCont>
      <vnsSvcRedirectPol name="N1Ext" AnycastEnabled="yes">
        <vnsRedirectDest ip="2000::25/128" mac="00:00:00:00:00:07"/>
      </vnsSvcRedirectPol>
      <vnsSvcRedirectPol name="N1Int" AnycastEnabled="yes">
        <vnsRedirectDest ip="30.30.30.100/32" mac="00:00:00:00:00:08"/>
      </vnsSvcRedirectPol>
    </vnsSvcCont>
  </fvTenant>
</polUni>
```

Step 3 To configure Anycast services as part of a Layer 4 to Layer 7 service graph without PBR, send a post with XML such as the following example:

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- /api/policymgr/mo/.xml -->
<polUni>
  <fvTenant name="tn1" >
    <vnsLDevCtx ctrctNameOrLbl="webCtrct" graphNameOrLbl="WebGraph" nodeNameOrLbl="N1">
      <vnsRsLDevCtxToLDev tDn="uni/tn-tn1/lDevVip-N1"/>
      <vnsLIfCtx connNameOrLbl="provider">
        <fvSubnet ip="50.50.50.50/32" ctrl="no-default-gateway">
          <fvEpAnycast mac="00:00:00:00:00:50"/>
        </fvSubnet>
        <vnsRsLIfCtxToBD tDn="uni/tn-coke/BD-N1IntBD"/>
        <vnsRsLIfCtxToLIf tDn="uni/tn-coke/lDevVip-N1/lIf-internal"/>
      </vnsLIfCtx>
      <vnsLIfCtx connNameOrLbl="consumer">
        <fvSubnet ip="2000::25/128" ctrl="no-default-gateway">
          <fvEpAnycast mac="00:00:00:00:00:51"/>
        </fvSubnet>
        <vnsRsLIfCtxToBD tDn="uni/tn-coke/BD-N1ExtBD"/>
        <vnsRsLIfCtxToLIf tDn="uni/tn-coke/lDevVip-N1/lIf-external"/>
      </vnsLIfCtx>
    </vnsLDevCtx>
  </fvTenant>
</polUni>
```



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

Remote Leaf Switches

About Remote Leaf Switches in the ACI Fabric

With an ACI fabric deployed, you can extend ACI services and APIC management to remote data centers with Cisco ACI leaf switches that have no local spine switch or APIC attached.

The remote leaf switches are added to an existing pod in the fabric. All policies deployed in the main data center are deployed in the remote switches, which behave like local leaf switches belonging to the pod. In this topology, all unicast traffic is through VXLAN over Layer 3. Layer 2 broadcast, unknown unicast, and multicast (BUM) messages are sent using Head End Replication (HER) tunnels without the use of Layer 3 multicast (bidirectional PIM) over the WAN. Any traffic that requires use of the spine switch proxy is forwarded to the main data center.

The APIC system discovers the remote leaf switches when they come up. From that time, they can be managed through APIC, as part of the fabric.



Note

- All inter-VRF traffic (pre-release 4.0(1)) goes to the spine switch before being forwarded.
 - For releases prior to Release 4.1(2), before decommissioning a remote leaf switch, you must first delete the vPC.
-

Characteristics of Remote Leaf Switch Behavior in Release 4.0(1)

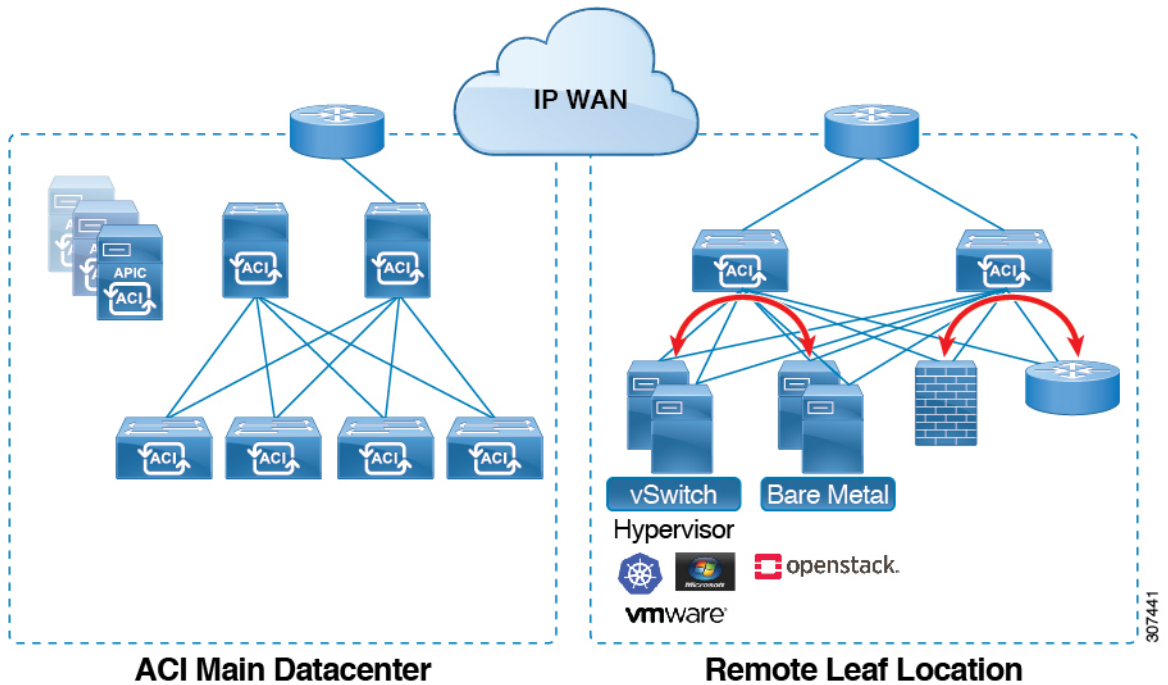
Starting in Release 4.0(1), Remote Leaf behavior takes on the following characteristics:

- Reduction of WAN bandwidth use by decoupling services from spine-proxy:
 - PBR: For local PBR devices or PBR devices behind a vPC, local switching is used without going to the spine proxy. For PBR devices on orphan ports on a peer remote leaf, a RL-vPC tunnel is used. This is true when the spine link to the main DC is functional or not functional.
 - ERSPAN: For peer destination EPGs, a RL-vPC tunnel is used. EPGs on local orphan or vPC ports use local switching to the destination EPG. This is true when the spine link to the main DC is functional or not functional.
 - Shared Services: Packets do not use spine-proxy path reducing WAN bandwidth consumption.
 - Inter-VRF traffic is forwarded through an upstream router and not placed on the spine.
 - This enhancement is only applicable for a remote leaf vPC pair. For communication across remote leaf pairs, a spine proxy is still used.
- Resolution of unknown L3 endpoints (through ToR glean process) in a remote leaf location when spine-proxy is not reachable.

Characteristics of Remote Leaf Switch Behavior in Release 4.1(2)

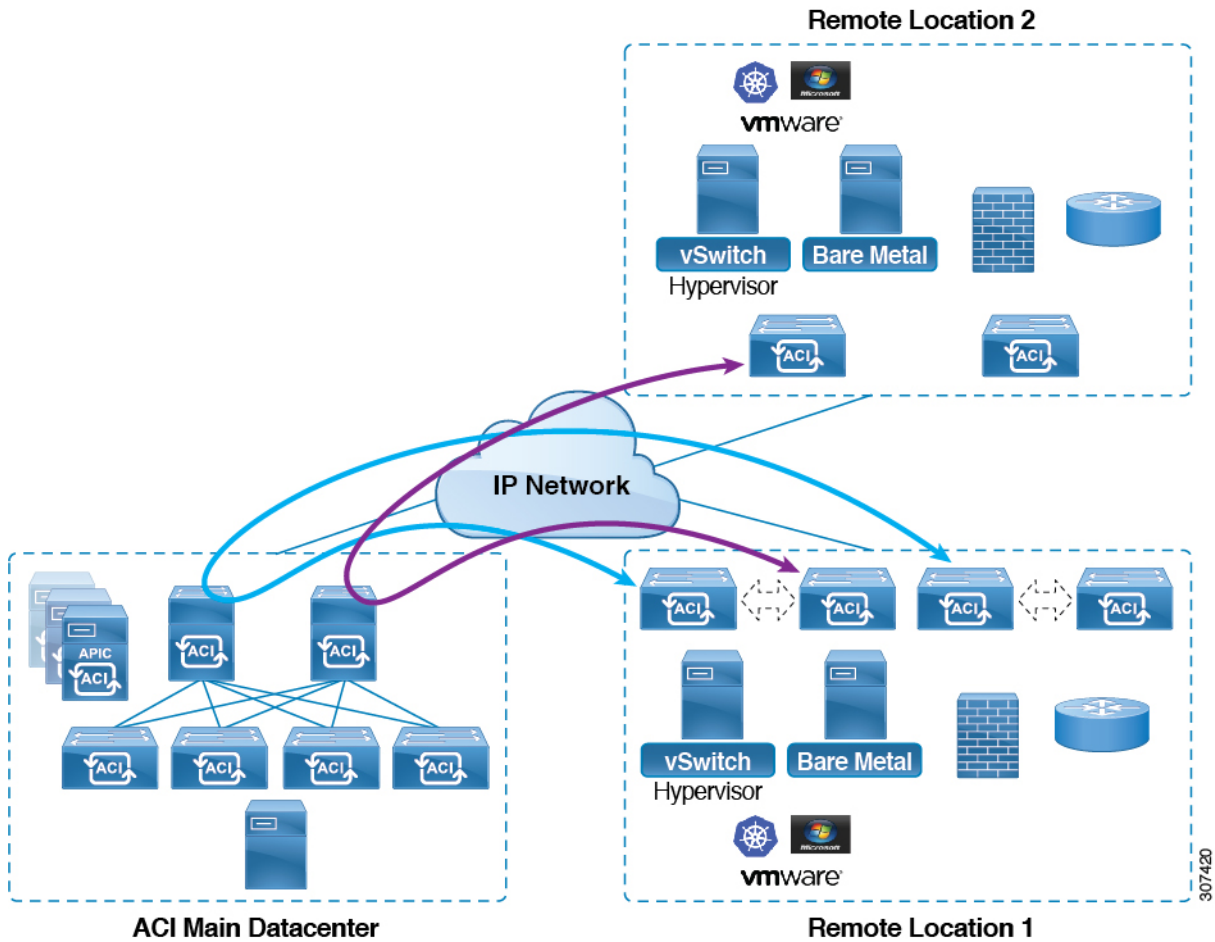
Before Release 4.1(2), all local switching (within the remote leaf vPC peer) traffic on the remote leaf location is switched directly between endpoints, whether physical or virtual, as shown in the following figure.

Figure 38: Local Switching Traffic: Prior to Release 4.1(2)



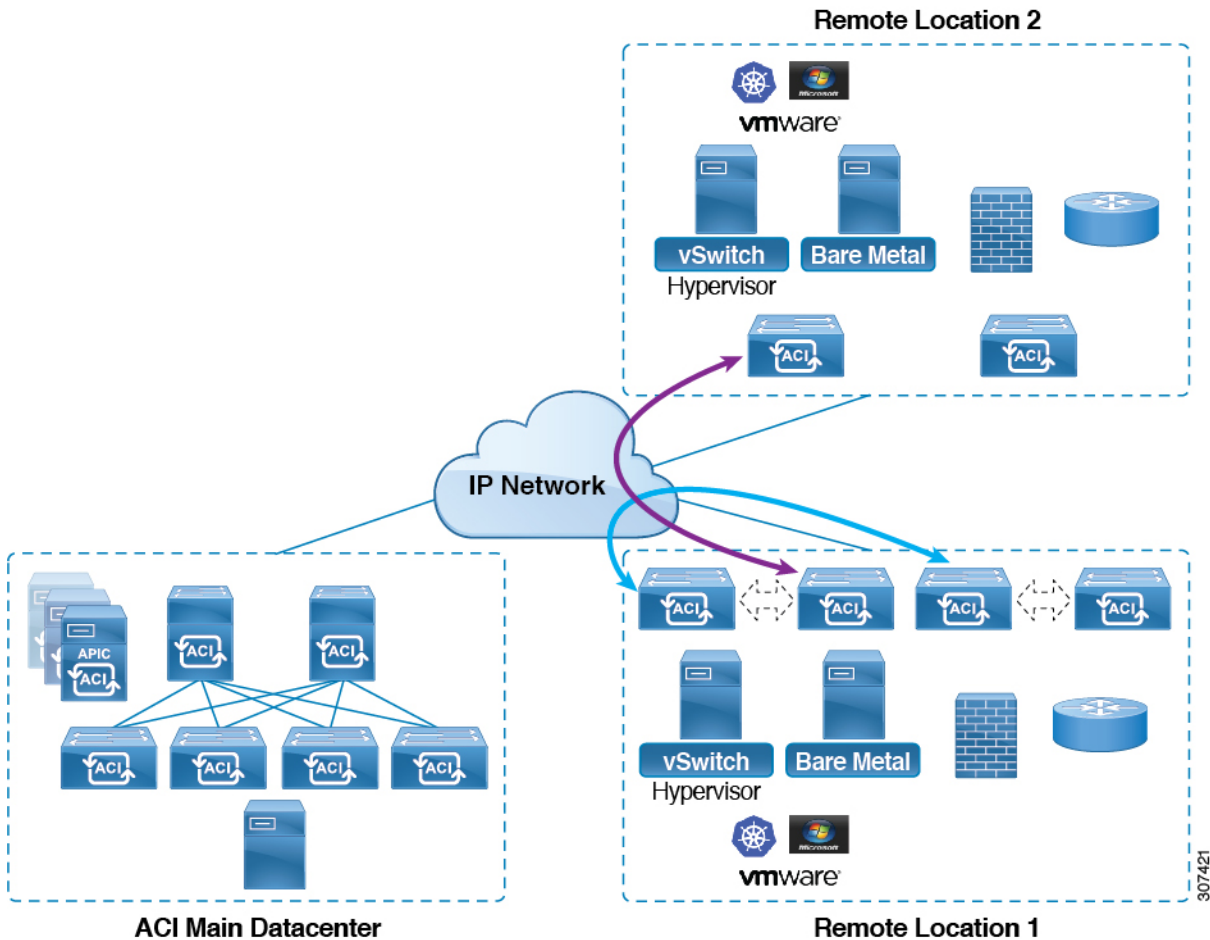
In addition, before Release 4.1(2), traffic between the remote leaf switch vPC pairs, either within a remote location or between remote locations, is forwarded to the spine switches in the ACI main data center pod, as shown in the following figure.

Figure 39: Remote Switching Traffic: Prior to Release 4.1(2)



Starting in Release 4.1(2), support is now available for direct traffic forwarding between remote leaf switches in different remote locations. This functionality offers a level of redundancy and availability in the connections between remote locations, as shown in the following figure.

Figure 40: Remote Leaf Switch Behavior: Release 4.1(2)



In addition, remote leaf switch behavior also takes on the following characteristics starting in release 4.1(2):

- Starting with Release 4.1(2), with direct traffic forwarding, when a spine switch fails within a single-pod configuration, the following occurs:
 - Local switching will continue to function for existing and new end point traffic between the remote leaf switch vPC peers, as shown in the "Local Switching Traffic: Prior to Release 4.1(2)" figure above.
 - For traffic between remote leaf switches across remote locations:
 - New end point traffic will fail because the remote leaf switch-to-spine switch tunnel would be down. From the remote leaf switch, new end point details will not get synced to the spine switch, so the other remote leaf switch pairs in the same or different locations cannot download the new end point information from COOP.
 - For uni-directional traffic, existing remote end points will age out after 300 secs, so traffic will fail after that point. Bi-directional traffic within a remote leaf site (between remote leaf VPC pairs) in a pod will get refreshed and will continue to function. Note that Bi-directional traffic to remote locations (remote leaf switches) will be affected as the remote end points will be expired by COOP after a timeout of 900 seconds.

- Bi-directional traffic within a remote leaf site (between remote leaf VPC pairs) in a pod will get refreshed and will continue to function. Note that Bi-directional traffic to remote locations (remote leaf switches) will be affected as the remote end points will be expired by COOP after a timeout of 900 seconds.
 - For shared services (inter-VRF), bi-directional traffic between end points belonging to remote leaf switches attached to two different remote locations in the same pod will fail after the remote leaf switch COOP end point age-out time (900 sec). This is because the remote leaf switch-to-spine COOP session would be down in this situation. However, shared services traffic between end points belonging to remote leaf switches attached to two different pods will fail after 30 seconds, which is the COOP fast-aging time.
 - L3Out-to-L3Out communication would not be able to continue because the BGP session to the spine switches would be down.
-
- When there is remote leaf direct uni-directional traffic, where the traffic is from remote leaf switch to remote leaf switch or from remote leaf switch to local leaf switch, there will be a milli-second traffic loss every time the remote end point (XR EP) timeout of 300 seconds occurs.

You can configure Remote Leaf in the APIC GUI, either with and without a wizard, or use the REST API or the NX-OS style CLI.

Remote Leaf Switch Hardware Requirements

The following switches are supported for the Remote Leaf Switch feature.

Fabric Spine Switches

For the spine switch at the ACI Main Datacenter that is connected to the WAN router, the following spine switches are supported:

- Fixed spine switches Cisco Nexus 9000 series:
 - N9K-C9316D-GX
 - N9K-C9332C
 - N9K-C9364C
 - N9K-C9364C-GX
- For modular spine switches, only Cisco Nexus 9000 series switches with names that end in EX, and later (for example, N9K-X9732C- *EX*) are supported.
- Older generation spine switches, such as the fixed spine switch N9K-C9336PQ or modular spine switches with the N9K-X9736PQ linecard are supported in the Main Datacenter, but only next generation spine switches are supported to connect to the WAN.

Remote Leaf Switches

- For the remote leaf switches, only Cisco Nexus 9000 series switches with names that end in EX, and later (for example, N9K-C93180LC-EX) are supported.

- The remote leaf switches must be running a switch image of 13.1.x or later (aci-n9000-dk9.13.1.x.x.bin) before they can be discovered. This may require manual upgrades on the leaf switches.

Remote Leaf Switch Restrictions and Limitations

The following guidelines and restrictions apply to remote leaf switches:

- A remote leaf vPC pair has a split brain condition when the DP-TEP address of one of the switches is not reachable from the peer. In this case, both remote leaf switches are up and active in the fabric and the COOP session is also up on both of the peers. One of the remote leaf switches does not have a route to the DP-TEP address of its peer, and due to this, the vPC has a split brain condition. Both of the node roles is changed to "primary" and all the front panel links are up in both of the peers while the zero message queue (ZMQ) session is down.
- The remote leaf solution requires the /32 tunnel end point (TEP) IP addresses of the remote leaf switches and main data center leaf/spine switches to be advertised across the main data center and remote leaf switches without summarization.
- If you move a remote leaf switch to a different site within the same pod and the new site has the same node ID as the original site, you must delete and recreate the virtual port channel (vPC).
- With the Cisco N9K-C9348GC-FXP switch, you can perform the initial remote leaf switch discovery only on ports 1/53 or 1/54. Afterward, you can use the other ports for fabric uplinks to the ISN/IPN for the remote leaf switch.

The following sections provide information on what is supported and not supported with remote leaf switches:

- [Supported Features, on page 340](#)
- [Unsupported Features, on page 341](#)

Supported Features

Beginning with Cisco APIC release 4.1(2), the following features are supported:

- Remote leaf switches with ACI Multi-Site
- Traffic forwarding directly across two remote leaf vPC pairs in the same remote data center or across data centers, when those remote leaf pairs are associated to the same pod or to pods that are part of the same multipod fabric
- Transit L3Out across remote locations, which is when the main Cisco ACI data center pod is a transit between two remote locations (the L3Out in RL location-1 and L3Out in RL location-2 are advertising prefixes for each other)

Beginning with Cisco APIC release 4.0(1), the following features are supported:

- Q-in-Q Encapsulation Mapping for EPGs
- PBR Tracking on remote leaf switches (with system-level global GIPo enabled)
- PBR Resilient Hashing
- Netflow
- MacSec Encryption

- Troubleshooting Wizard
- Atomic counters

Unsupported Features

Full fabric and tenant policies are supported on remote leaf switches in this release with the exception of the following features, which are unsupported:

- GOLF
- vPod
- Floating L3Out
- Fast-convergence mode
- Stretching of L3Out SVI between local leaf switches (ACI main data center switches) and remote leaf switches or stretching across two different vPC pairs of remote leaf switches
- Copy service is not supported when deployed on local leaf switches and when the source or destination is on the remote leaf switch. In this situation, the routable TEP IP address is not allocated for the local leaf switch. For more information, see the section "Copy Services Limitations" in the "Configuring Copy Services" chapter in the *Cisco APIC Layer 4 to Layer 7 Services Deployment Guide*, available in the [APIC documentation page](#).
- Layer 2 Outside Connections (except Static EPGs)
- 802.1Q Tunnels
- Copy services with vzAny contract
- FCoE connections on remote leaf switches
- Flood in encapsulation for bridge domains or EPGs
- Fast Link Failover policies
- Managed Service Graph-attached devices at remote locations
- Traffic Storm Control
- Cloud Sec Encryption
- First Hop Security
- Layer 3 Multicast routing on remote leaf switches
- Maintenance mode
- TEP to TEP atomic counters

The following scenarios are not supported when integrating remote leaf switches in a Multi-Site architecture in conjunction with the intersite L3Out functionality:

- Transit routing between L3Outs deployed on remote leaf switch pairs associated to separate sites
- Endpoints connected to a remote leaf switch pair associated to a site communicating with the L3Out deployed on the remote leaf switch pair associated to a remote site

- Endpoints connected to the local site communicating with the L3Out deployed on the remote leaf switch pair associated to a remote site
- Endpoints connected to a remote leaf switch pair associated to a site communicating with the L3Out deployed on a remote site



Note The limitations above do not apply if the different data center sites are deployed as pods as part of the same Multi-Pod fabric.

The following deployments and configurations are not supported with the remote leaf switch feature:

- It is not supported to stretch a bridge domain between remote leaf nodes associated to a given site (APIC domain) and leaf nodes part of a separate site of a Multi-Site deployment (in both scenarios where those leaf nodes are local or remote) and a fault is generated on APIC to highlight this restriction. This applies independently from the fact that BUM flooding is enabled or disabled when configuring the stretched bridge domain on the Multi-Site Orchestrator (MSO). However, a bridge domain can always be stretched (with BUM flooding enabled or disabled) between remote leaf nodes and local leaf nodes belonging to the same site (APIC domain).
- Spanning Tree Protocol across remote leaf location and main data center
- APICs directly connected to remote leaf switches
- Orphan port channel or physical ports on remote leaf switches, with a vPC domain (this restriction applies for releases 3.1 and earlier)
- With and without service node integration, local traffic forwarding within a remote location is only supported if the consumer, provider, and services nodes are all connected to remote leaf switches are in vPC mode
- /32 loopbacks advertised from the spine switch to the IPN must not be suppressed/aggregated toward the remote leaf switch. The /32 loopbacks must be advertised to the remote leaf switch.

WAN Router and Remote Leaf Switch Configuration Guidelines

Before a remote leaf is discovered and incorporated in APIC management, you must configure the WAN router and the remote leaf switches.

Configure the WAN routers that connect to the fabric spine switch external interfaces and the remote leaf switch ports, with the following requirements:

WAN Routers

- Enable OSPF on the interfaces, with the same details, such as area ID, type, and cost.
- Configure DHCP Relay on the interface leading to each APIC's IP address in the main fabric.
- The interfaces on the WAN routers which connect to the VLAN-5 interfaces on the spine switches must be on different VRFs than the interfaces connecting to a regular multipod network.

Remote Leaf Switches

- Connect the remote leaf switches to an upstream router by a direct connection from one of the fabric ports. The following connections to the upstream router are supported:
 - 40 Gbps & higher connections
 - With a QSFP-to-SFP Adapter, supported 1G/10G SFPs

Bandwidth in the WAN must be a minimum of 100 Mbps and maximum supported latency is 300 msec.

- It is recommended, but not required to connect the pair of remote leaf switches with a vPC. The switches on both ends of the vPC must be remote leaf switches at the same remote datacenter.
- Configure the northbound interfaces as Layer 3 sub-interfaces on VLAN-4, with unique IP addresses. If you connect more than one interface from the remote leaf switch to the router, configure each interface with a unique IP address.
- Enable OSPF on the interfaces, but do not set the OSPF area type as stub area.
- The IP addresses in the remote leaf switch TEP Pool subnet must not overlap with the pod TEP subnet pool. The subnet used must be /24 or lower.
- Multipod is supported, but not required, with the Remote Leaf feature.
- When connecting a pod in a single-pod fabric with remote leaf switches, configure an L3Out from a spine switch to the WAN router and an L3Out from a remote leaf switch to the WAN router, both using VLAN-4 on the switch interfaces.
- When connecting a pod in a multipod fabric with remote leaf switches, configure an L3Out from a spine switch to the WAN router and an L3Out from a remote leaf switch to the WAN router, both using VLAN-4 on the switch interfaces. Also configure a multipod-internal L3Out using VLAN-5 to support traffic that crosses pods destined to a remote leaf switch. The regular multipod and multipod-internal connections can be configured on the same physical interfaces, as long as they use VLAN-4 and VLAN-5.
- When configuring the Multipod-internal L3Out, use the same router ID as for the regular multipod L3Out, but deselect the **Use Router ID as Loopback Address** option for the router-id and configure a different loopback IP address. This enables ECMP to function.

Configure Remote Leaf Switches Using the REST API

To enable Cisco APIC to discover and connect the IPN router and remote leaf switches, perform the steps in this topic.

This example assumes that the remote leaf switches are connected to a pod in a multipod topology. It includes two L3Outs configured in the infra tenant, with VRF overlay-1:

- One is configured on VLAN-4, that is required for both the remote leaf switches and the spine switch that is connected to the WAN router.
- One is the multipod-internal L3Out configured on VLAN-5, that is required for the multipod and Remote Leaf features, when they are deployed together.

Step 1

To define the TEP pool for two remote leaf switches to be connected to a pod, send a post with XML such as the following example:

Example:

```
<fabricSetupPol>
  <fabricSetupP tepPool="10.0.0.0/16" podId="1" >
    <fabricExtSetupP tepPool="30.0.128.0/20" extPoolId="1"/>
  </fabricSetupP>
  <fabricSetupP tepPool="10.1.0.0/16" podId="2" >
    <fabricExtSetupP tepPool="30.1.128.0/20" extPoolId="1"/>
  </fabricSetupP>
</fabricSetupPol>
```

Step 2 To define the node identity policy, send a post with XML, such as the following example:

Example:

```
<fabricNodeIdentPol>
  <fabricNodeIdentP serial="SAL17267Z7W" name="leaf1" nodeId="101" podId="1"
extPoolId="1" nodeType="remote-leaf-wan"/>
  <fabricNodeIdentP serial="SAL17267Z7X" name="leaf2" nodeId="102" podId="1"
extPoolId="1" nodeType="remote-leaf-wan"/>
  <fabricNodeIdentP serial="SAL17267Z7Y" name="leaf3" nodeId="201" podId="1"
extPoolId="1" nodeType="remote-leaf-wan"/>
  <fabricNodeIdentP serial="SAL17267Z7Z" name="leaf4" nodeId="201" podId="1"
extPoolId="1" nodeType="remote-leaf-wan"/>
</fabricNodeIdentPol>
```

Step 3 To configure the Fabric External Connection Profile, send a post with XML such as the following example:

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<imdata totalCount="1">
  <fvFabricExtConnP dn="uni/tn-infra/fabricExtConnP-1" id="1" name="Fabric_Ext_Conn_Poll1"
rt="extended:as2-nn4:5:16" siteId="0">
    <l3extFabricExtRoutingP name="test">
      <l3extSubnet ip="150.1.0.0/16" scope="import-security"/>
    </l3extFabricExtRoutingP>
    <l3extFabricExtRoutingP name="ext_routing_prof_1">
      <l3extSubnet ip="204.1.0.0/16" scope="import-security"/>
      <l3extSubnet ip="209.2.0.0/16" scope="import-security"/>
      <l3extSubnet ip="202.1.0.0/16" scope="import-security"/>
      <l3extSubnet ip="207.1.0.0/16" scope="import-security"/>
      <l3extSubnet ip="200.0.0.0/8" scope="import-security"/>
      <l3extSubnet ip="201.2.0.0/16" scope="import-security"/>
      <l3extSubnet ip="210.2.0.0/16" scope="import-security"/>
      <l3extSubnet ip="209.1.0.0/16" scope="import-security"/>
      <l3extSubnet ip="203.2.0.0/16" scope="import-security"/>
      <l3extSubnet ip="208.1.0.0/16" scope="import-security"/>
      <l3extSubnet ip="207.2.0.0/16" scope="import-security"/>
      <l3extSubnet ip="100.0.0.0/8" scope="import-security"/>
      <l3extSubnet ip="201.1.0.0/16" scope="import-security"/>
      <l3extSubnet ip="210.1.0.0/16" scope="import-security"/>
      <l3extSubnet ip="203.1.0.0/16" scope="import-security"/>
      <l3extSubnet ip="208.2.0.0/16" scope="import-security"/>
    </l3extFabricExtRoutingP>
    <fvPodConnP id="1">
      <fvIp addr="100.11.1.1/32"/>
    </fvPodConnP>
    <fvPodConnP id="2">
      <fvIp addr="200.11.1.1/32"/>
    </fvPodConnP>
    <fvPeeringP type="automatic_with_full_mesh"/>
  </fvFabricExtConnP>
</imdata>
```

Step 4 To configure an L3Out on VLAN-4, that is required for both the remote leaf switches and the spine switch connected to the WAN router, enter XML such as the following example.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<polUni>
<fvTenant name="infra">
  <l3extOut name="rleaf-wan-test">
    <ospfExtP areaId="0.0.0.5"/>
    <bgpExtP/>
    <l3extRsEctx tnFvCtxName="overlay-1"/>
    <l3extRsL3DomAtt tDn="uni/l3dom-l3extDom1"/>
    <l3extProvLbl descr="" name="prov_mp1" ownerKey="" ownerTag="" tag="yellow-green"/>
    <l3extLNodeP name="rleaf-101">
      <l3extRsNodeL3OutAtt rtrId="202.202.202.202" tDn="topology/pod-1/node-101">
        </l3extRsNodeL3OutAtt>
        <l3extLIIfP name="portIf">
          <l3extRsPathL3OutAtt ifInstT="sub-interface" tDn="topology/pod-1/paths-101/pathep-[eth1/49]"
            addr="202.1.1.2/30" mac="AA:11:22:33:44:66" encap='vlan-4'/>
          <ospfIfP>
            <ospfRsIfPol tnOspfIfPolName='ospfIfPol'/>
          </ospfIfP>
        </l3extLIIfP>
      </l3extLNodeP>
      <l3extLNodeP name="r1Spine-201">
        <l3extRsNodeL3OutAtt rtrId="201.201.201.201" rtrIdLoopBack="no" tDn="topology/pod-1/node-201">
          <!--
          <l3extLoopBackIfP addr="201::201/128" descr="" name=""/>
          <l3extLoopBackIfP addr="201.201.201.201/32" descr="" name=""/>
          -->
          <l3extLoopBackIfP addr="::" />
        </l3extRsNodeL3OutAtt>
        <l3extLIIfP name="portIf">
          <l3extRsPathL3OutAtt ifInstT="sub-interface" tDn="topology/pod-1/paths-201/pathep-[eth8/36]"
            addr="201.1.1.1/30" mac="00:11:22:33:77:55" encap='vlan-4'/>
          <ospfIfP>
            <ospfRsIfPol tnOspfIfPolName='ospfIfPol'/>
          </ospfIfP>
        </l3extLIIfP>
      </l3extLNodeP>
      <l3extInstP descr="" matchT="AtleastOne" name="instpl" prio="unspecified" targetDscp="unspecified">
        <fvRsCustQosPol tnQosCustomPolName=""/>
      </l3extInstP>
    </l3extOut>
    <ospfIfPol name="ospfIfPol" nwT="bcast"/>
  </fvTenant>
</polUni>
```

Step 5 For releases prior to Release 4.1(2), to configure the multipod L3Out on VLAN-5, that is required for both multipod and the remote leaf topology, send a post such as the following example.

Note Do not enter this information if you are deploying new remote leaf switches running Release 4.1(2) or later and you are enabling direct traffic forwarding on those remote leaf switches. Configuring an OSPF instance using VLAN-5 for multipod is not needed in this case.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<polUni>

  <fvTenant name="infra" >
```

```

<l3extOut name="ipn-multipodInternal">
  <ospfExtP areaCtrl="inherit-ipsec,redistribute,summary" areaId="0.0.0.5" multipodInternal="yes"
/>
  <l3extRsEctx tnFvCtxName="overlay-1" />
  <l3extLNodeP name="bLeaf">
    <l3extRsNodeL3OutAtt rtrId="202.202.202.202" rtrIdLoopBack="no" tDn="topology/pod-2/node-202">

      <l3extLoopBackIfP addr="202.202.202.212"/>
    </l3extRsNodeL3OutAtt>
    <l3extRsNodeL3OutAtt rtrId="102.102.102.102" rtrIdLoopBack="no" tDn="topology/pod-1/node-102">

      <l3extLoopBackIfP addr="102.102.102.112"/>
    </l3extRsNodeL3OutAtt>
    <l3extLIfP name="portIf">
      <ospfIfP authKeyId="1" authType="none">
        <ospfRsIfPol tnOspfIfPolName="ospfIfPol" />
      </ospfIfP>
      <l3extRsPathL3OutAtt addr="10.0.254.233/30" encap="vlan-5" ifInstT="sub-interface"
tDn="topology/pod-2/paths-202/pathep-[eth5/2]"/>
      <l3extRsPathL3OutAtt addr="10.0.255.229/30" encap="vlan-5" ifInstT="sub-interface"
tDn="topology/pod-1/paths-102/pathep-[eth5/2]"/>
    </l3extLIfP>
  </l3extLNodeP>
  <l3extInstP matchT="AtleastOne" name="ipnInstP" />
</l3extOut>
</fvTenant>
</polUni>

```

Prerequisites Required Prior to Downgrading Remote Leaf Switches



Note If you have remote leaf switches deployed, if you downgrade the APIC software from Release 3.1(1) or later, to an earlier release that does not support the Remote Leaf feature, you must decommission the remote nodes and remove the remote leaf-related policies (including the TEP Pool), before downgrading. For more information on decommissioning switches, see *Decommissioning and Recommissioning Switches* in the *Cisco APIC Troubleshooting Guide*.

Before you downgrade remote leaf switches, verify that the followings tasks are complete:

- Delete the vPC domain.
- Delete the vTEP - Virtual Network Adapter if using SCVMM.
- Decommission the remote leaf nodes, and wait 10 -15 minutes after the decommission for the task to complete.
- Delete the remote leaf to WAN L3out in the infra tenant.
- Delete the infra-l3out with VLAN 5 if using Multipod.
- Delete the remote TEP pools.

HSRP

About HSRP

HSRP is a first-hop redundancy protocol (FHRP) that allows a transparent failover of the first-hop IP router. HSRP provides first-hop routing redundancy for IP hosts on Ethernet networks configured with a default router IP address. You use HSRP in a group of routers for selecting an active router and a standby router. In a group of routers, the active router is the router that routes packets, and the standby router is the router that takes over when the active router fails or when preset conditions are met.

Many host implementations do not support any dynamic router discovery mechanisms but can be configured with a default router. Running a dynamic router discovery mechanism on every host is not practical for many reasons, including administrative overhead, processing overhead, and security issues. HSRP provides failover services to such hosts.

When you use HSRP, you configure the HSRP virtual IP address as the default router of the host (instead of the IP address of the actual router). The virtual IP address is an IPv4 or IPv6 address that is shared among a group of routers that run HSRP.

When you configure HSRP on a network segment, you provide a virtual MAC address and a virtual IP address for the HSRP group. You configure the same virtual address on each HSRP-enabled interface in the group. You also configure a unique IP address and MAC address on each interface that acts as the real address. HSRP selects one of these interfaces to be the active router. The active router receives and routes packets destined for the virtual MAC address of the group.

HSRP detects when the designated active router fails. At that point, a selected standby router assumes control of the virtual MAC and IP addresses of the HSRP group. HSRP also selects a new standby router at that time.

HSRP uses a priority designator to determine which HSRP-configured interface becomes the default active router. To configure an interface as the active router, you assign it with a priority that is higher than the priority of all the other HSRP-configured interfaces in the group. The default priority is 100, so if you configure just one interface with a higher priority, that interface becomes the default active router.

Interfaces that run HSRP send and receive multicast User Datagram Protocol (UDP)-based hello messages to detect a failure and to designate active and standby routers. When the active router fails to send a hello message within a configurable period of time, the standby router with the highest priority becomes the active router. The transition of packet forwarding functions between the active and standby router is completely transparent to all hosts on the network.

You can configure multiple HSRP groups on an interface. The virtual router does not physically exist but represents the common default router for interfaces that are configured to provide backup to each other. You do not need to configure the hosts on the LAN with the IP address of the active router. Instead, you configure them with the IP address of the virtual router (virtual IP address) as their default router. If the active router fails to send a hello message within the configurable period of time, the standby router takes over, responds to the virtual addresses, and becomes the active router, assuming the active router duties. From the host perspective, the virtual router remains the same.



Note Packets received on a routed port destined for the HSRP virtual IP address terminate on the local router, regardless of whether that router is the active HSRP router or the standby HSRP router. This process includes ping and Telnet traffic. Packets received on a Layer 2 (VLAN) interface destined for the HSRP virtual IP address terminate on the active router.

Guidelines and Limitations

Follow these guidelines and limitations:

- The HSRP state must be the same for both HSRP IPv4 and IPv6. The priority and preemption must be configured to result in the same state after failovers.
- Currently, only one IPv4 and one IPv6 group is supported on the same sub-interface in Cisco ACI. Even when dual stack is configured, Virtual MAC must be the same in IPv4 and IPv6 HSRP configurations.
- BFD IPv4 and IPv6 is supported when the network connecting the HSRP peers is a pure layer 2 network. You must configure a different router MAC address on the leaf switches. The BFD sessions become active only if you configure different MAC addresses in the leaf interfaces.
- Users must configure the same MAC address for IPv4 and IPv6 HSRP groups for dual stack configurations.
- HSRP VIP must be in the same subnet as the interface IP.
- It is recommended that you configure interface delay for HSRP configurations.
- HSRP is only supported on routed-interface or sub-interface. HSRP is not supported on VLAN interfaces and switched virtual interface (SVI). Therefore, no VPC support for HSRP is available.
- Object tracking on HSRP is not supported.
- HSRP Management Information Base (MIB) for SNMP is not supported.
- Multiple group optimization (MGO) is not supported with HSRP.
- ICMP IPv4 and IPv6 redirects are not supported.
- Cold Standby and Non-Stop Forwarding (NSF) are not supported because HSRP cannot be restarted in the Cisco ACI environment.
- There is no extended hold-down timer support as HSRP is supported only on leaf switches. HSRP is not supported on spine switches.
- HSRP version change is not supported in APIC. You must remove the configuration and reconfigure with the new version.
- HSRP version 2 does not inter-operate with HSRP version 1. An interface cannot operate both version 1 and version 2 because both versions are mutually exclusive. However, the different versions can be run on different physical interfaces of the same router.
- Route Segmentation is programmed in Cisco Nexus 93128TX, Cisco Nexus 9396PX, and Cisco Nexus 9396TX leaf switches when HSRP is active on the interface. Therefore, there is no DMAC=router MAC check conducted for route packets on the interface. This limitation does not apply for Cisco Nexus 93180LC-EX, Cisco Nexus 93180YC-EX, and Cisco Nexus 93108TC-EX leaf switches.

- HSRP configurations are not supported in the Basic GUI mode. The Basic GUI mode has been deprecated starting with APIC release 3.0(1).
- Fabric to Layer 3 Out traffic will always load balance across all the HSRP leaf switches, irrespective of their state. If HSRP leaf switches span multiple pods, the fabric to out traffic will always use leaf switches in the same pod.
- This limitation applies to some of the earlier Cisco Nexus 93128TX, Cisco Nexus 9396PX, and Cisco Nexus 9396TX switches. When using HSRP, the MAC address for one of the routed interfaces or routed sub-interfaces must be modified to prevent MAC address flapping on the Layer 2 external device. This is because Cisco APIC assigns the same MAC address (00:22:BD:F8:19:FF) to every logical interface under the interface logical profiles.

Configuring HSRP in APIC Using REST API

HSRP is enabled when the leaf switch is configured.

Before you begin

- The tenant and VRF must be configured.
- VLAN pools must be configured with the appropriate VLAN range defined and the appropriate Layer 3 domain created and attached to the VLAN pool.
- The Attach Entity Profile must also be associated with the Layer 3 domain.
- The interface profile for the leaf switches must be configured as required.

Step 1 Create port selectors.

Example:

```
<polUni>
  <infraInfra dn="uni/infra">
    <infraNodeP name="TenantNode_101">
      <infraLeafS name="leafselector" type="range">
        <infraNodeBlk name="nodeblk" from_"101" to_"101">
          </infraNodeBlk>
        </infraLeafS>
      <infraRsAccPortP tDn="uni/infra/accportprof-TenantPorts_101"/>
    </infraNodeP>
    <infraAccPortP name="TenantPorts_101">
      <infraHPortS name="portselector" type="range">
        <infraPortBlk name="portblk" fromCard="1" toCard="1" fromPort="41" toPort="41">
          </infraPortBlk>
        <infraRsAccBaseGrp tDn="uni/infra/funcprof/accportgrp-TenantPortGrp_101"/>
      </infraHPortS>
    </infraAccPortP>
    <infraFuncP>
      <infraAccPortGrp name="TenantPortGrp_101">
        <infraRsAttEntP tDn="uni/infra/attentp-AttEntityProfTenant"/>
        <infraRsHIfPol tnFabricHIfPolName="default"/>
      </infraAccPortGrp>
    </infraFuncP>
  </infraInfra>
</polUni>
```

Step 2 Create a tenant policy.**Example:**

```
<polUni>
  <fvTenant name="t9" dn="uni/tn-t9" descr="">
    <fvCtx name="t9_ctx1" pcEnfPref="unenforced">
      </fvCtx>
    <fvBD name="t9_bd1" unkMacUcastAct="flood" arpFlood="yes">
      <fvRsCtx tnFvCtxName="t9_ctx1"/>
      <fvSubnet ip="101.9.1.1/24" scope="shared"/>
    </fvBD>
    <l3extOut dn="uni/tn-t9/out-l3extOut1" enforceRtctrl="export" name="l3extOut1">
      <l3extLNodeP name="Node101">
        <l3extRsNodeL3OutAtt rtrId="210.210.121.121" rtrIdLoopBack="no" tDn="topology/pod-1/node-101"/>

        </l3extLNodeP>
        <l3extRsEctx tnFvCtxName="t9_ctx1"/>
        <l3extRsL3DomAtt tDn="uni/l3dom-dom1"/>
        <l3extInstP matchT="AtleastOne" name="extEpg" prio="unspecified" targetDscp="unspecified">
          <l3extSubnet aggregate="" descr="" ip="176.21.21.21/21" name="" scope="import-security"/>
        </l3extInstP>
      </l3extOut>
    </fvTenant>
  </polUni>
```

Step 3 Create an HSRP interface policy.**Example:**

```
<polUni>
  <fvTenant name="t9" dn="uni/tn-t9" descr="">
    <hsrpIfPol name="hsrpIfPol" ctrl="bfd" delay="4" reloadDelay="11"/>
  </fvTenant>
</polUni>
```

Step 4 Create an HSRP group policy.**Example:**

```
<polUni>
  <fvTenant name="t9" dn="uni/tn-t9" descr="">
    <hsrpIfPol name="hsrpIfPol" ctrl="bfd" delay="4" reloadDelay="11"/>
  </fvTenant>
</polUni>
```

Step 5 Create an HSRP interface profile and an HSRP group profile.**Example:**

```
<polUni>
  <fvTenant name="t9" dn="uni/tn-t9" descr="">
    <l3extOut dn="uni/tn-t9/out-l3extOut1" enforceRtctrl="export" name="l3extOut1">
      <l3extLNodeP name="Node101">
        <l3extLIfP name="eth1-41-v6" ownerKey="" ownerTag="" tag="yellow-green">
          <hsrpIfP name="eth1-41-v6" version="v2">
            <hsrpRsIfPol tnHsrpIfPolName="hsrpIfPol"/>
            <hsrpGroupP descr="" name="HSRPV6-2" groupId="330" groupAf="ipv6" ip="fe80::3"
mac="00:00:0C:18:AC:01" ipObtainMode="admin">
              <hsrpRsGroupPol tnHsrpGroupPolName="G1"/>
            </hsrpGroupP>
          </hsrpIfP>
          <l3extRsPathL3OutAtt addr="2002::100/64" descr="" encap="unknown" encapScope="local"
ifInstT="l3-port" llAddr="::" mac="00:22:BD:F8:19:FF" mode="regular" mtu="inherit"
tDn="topology/pod-1/paths-101/pathep-[eth1/41]" targetDscp="unspecified">
        </l3extLNodeP>
      </l3extOut>
    </fvTenant>
  </polUni>
```



```

        <l3extIp addr="2004::100/64"/>
      </l3extRsPathL3OutAtt>
    </l3extLIIfP>
    <l3extLIIfP name="eth1-41-v4" ownerKey="" ownerTag="" tag="yellow-green">
      <hsrpIfP name="eth1-41-v4" version="v1">
        <hsrpRsIfPol tnHsrpIfPolName="hsrpIfPol"/>
        <hsrpGroupP descr="" name="HSRPV4-2" groupId="51" groupAf="ipv4" ip="177.21.21.21"
mac="00:00:0C:18:AC:01" ipObtainMode="admin">
          <hsrpRsGroupPol tnHsrpGroupPolName="G1"/>
        </hsrpGroupP>
      </hsrpIfP>
      <l3extRsPathL3OutAtt addr="177.21.21.11/24" descr="" encap="unknown" encapScope="local"
ifInstT="l3-port" llAddr="::" mac="00:22:BD:F8:19:FF" mode="regular" mtu="inherit"
tDn="topology/pod-1/paths-101/pathep-[eth1/41]" targetDscp="unspecified">
        <l3extIp addr="177.21.23.11/24"/>
      </l3extRsPathL3OutAtt>
    </l3extLIIfP>
  </l3extLNodeP>
</l3extOut>
</fvTenant>
</polUni>

```

IP Multicast

Tenant Routed Multicast

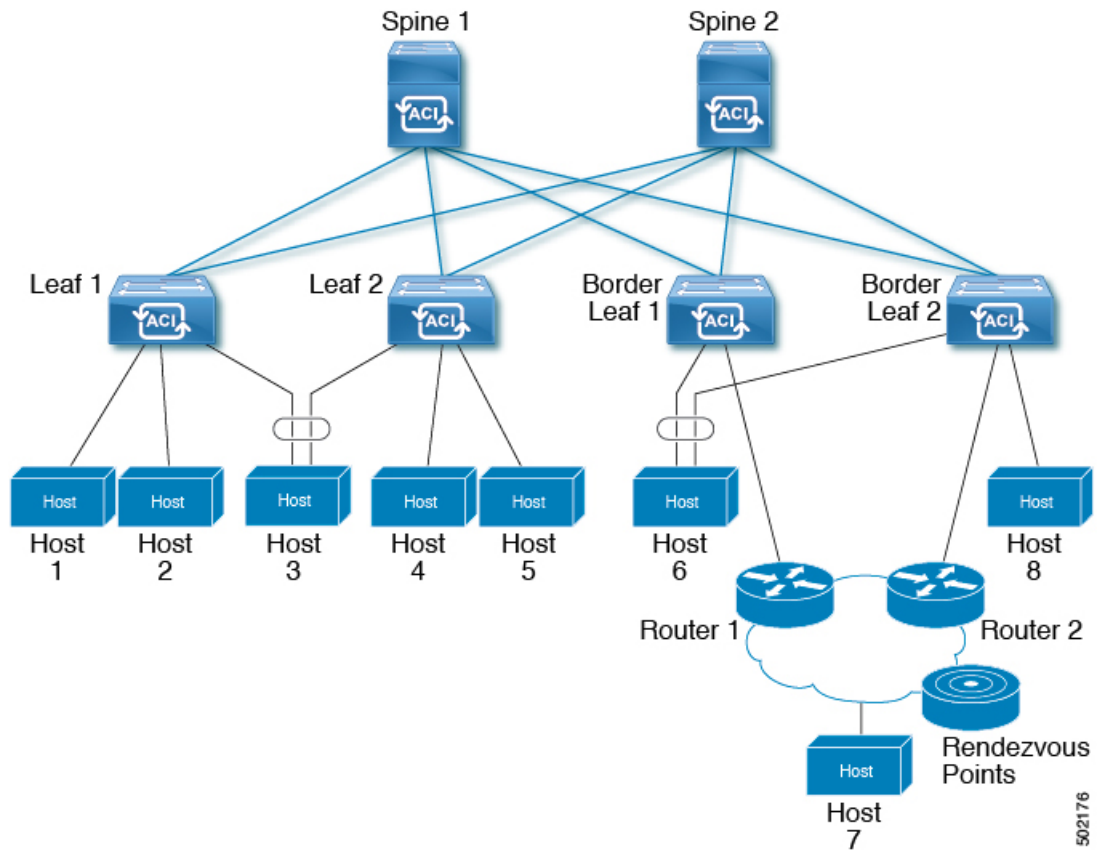
Cisco Application Centric Infrastructure (ACI) Tenant Routed Multicast (TRM) enables Layer 3 multicast routing in Cisco ACI tenant VRF instances. TRM supports multicast forwarding between senders and receivers within the same or different subnets. Multicast sources and receivers can be connected to the same or different leaf switches or external to the fabric using L3Out connections.

In the Cisco ACI fabric, most unicast and IPv4 multicast routing operate together on the same border leaf switches, with the IPv4 multicast protocol operating over the unicast routing protocols.

In this architecture, only the border leaf switches run the full Protocol Independent Multicast (PIM) protocol. Non-border leaf switches run PIM in a passive mode on the interfaces. They do not peer with any other PIM routers. The border leaf switches peer with other PIM routers connected to them over L3Outs and also with each other.

The following figure shows border leaf switch 1 and border leaf switch 2 connecting to router 1 and router 2 in the IPv4 multicast cloud. Each virtual routing and forwarding (VRF) instance in the fabric that requires IPv4 multicast routing will peer separately with external IPv4 multicast routers.

Figure 41: Overview of Multicast Cloud



Guidelines and Restrictions for Configuring Layer 3 Multicast

See the following guidelines and restrictions:

- Custom QoS policy is not supported for Layer 3 multicast traffic sourced from outside the ACI fabric (received from L3Out).
- Enabling PIMv4 (Protocol-Independent Multicast, version 4) and Advertise Host routes on a BD is not supported.
- If the border leaf switches in your ACI fabric are running multicast and you disable multicast on the L3Out while you still have unicast reachability, you will experience traffic loss if the external peer is a Cisco Nexus 9000 switch. This impacts cases where traffic is destined towards the fabric (where the sources are outside the fabric but the receivers are inside the fabric) or transiting through the fabric (where the source and receivers are outside the fabric, but the fabric is transit).
- If the (s, g) entry is installed on a border leaf switch, you might see drops in unicast traffic that comes from the fabric to this source outside the fabric when the following conditions are met:
 - Preferred group is used on the L3Out EPG
 - Unicast routing table for the source is using the default route 0.0.0.0/0

This behavior is expected.

- The Layer 3 multicast configuration is done at the VRF level so protocols function within the VRF and multicast is enabled in a VRF, and each multicast VRF can be turned on or off independently.
- Once a VRF is enabled for multicast, the individual bridge domains (BDs) and L3 Outs under the enabled VRF can be enabled for multicast configuration. By default, multicast is disabled in all BDs and Layer 3 Outs.
- Any Source Multicast (ASM) and Source-Specific Multicast (SSM) are supported.
- You can configure a maximum of four ranges for SSM multicast in the route map per VRF.
- Bidirectional PIM and PIM IPv6 are currently not supported.
- IGMP snooping cannot be disabled on pervasive bridge domains with multicast routing enabled.
- Multicast routers are not supported in pervasive bridge domains.
- The Layer 3 multicast feature is supported on the following leaf switches:
 - EX models:
 - N9K-93108TC-EX
 - N9K-93180LC-EX
 - N9K-93180YC-EX
 - FX models:
 - N9K-93108TC-FX
 - N9K-93180YC-FX
 - N9K-C9348GC-FXP
 - FX2 models:
 - N9K-93240YC-FX2
 - N9K-C9336C-FX2
- PIM is supported on Layer 3 Out routed interfaces and routed subinterfaces including Layer 3 port-channel interfaces. PIM is not supported on Layer 3 Out SVI interfaces.
- Enabling PIM on an L3Out causes an implicit external network to be configured. This action results in the L3Out being deployed and protocols potentially coming up even if you have not defined an external network.
- If the multicast source is connected to Leaf-A as an orphan port and you have an L3Out on Leaf-B, and Leaf-A and Leaf-B are in a vPC pair, the EPG encapsulation VLAN tied to the multicast source will need to be deployed on Leaf-B.
- For Layer 3 multicast support, when the ingress leaf switch receives a packet from a source that is attached on a bridge domain, and the bridge domain is enabled for multicast routing, the ingress leaf switch sends only a routed VRF copy to the fabric (routed implies that the TTL is decremented by 1, and the source-mac is rewritten with a pervasive subnet MAC). The egress leaf switch also routes the packet into receivers in all the relevant bridge domains. Therefore, if a receiver is on the same bridge domain as the source, but on a different leaf switch than the source, that receiver continues to get a routed copy, although it is

in the same bridge domain. This also applies if the source and receiver are on the same bridge domain and on the same leaf switch, if PIM is enabled on this bridge domain.

For more information, see details about Layer 3 multicast support for multipod that leverages existing Layer 2 design, at the following link [Adding Pods](#).

- Starting with release 3.1(1x), Layer 3 multicast is supported with FEX. Multicast sources or receivers that are connected to FEX ports are supported. For further details about how to add FEX in your testbed, see [Configure a Fabric Extender with Application Centric Infrastructure](https://www.cisco.com/c/en/us/support/docs/cloud-systems-management/application-policy-infrastructure-controller-apic/200529-Configure-a-Fabric-Extender-with-Applica.html) at this URL: <https://www.cisco.com/c/en/us/support/docs/cloud-systems-management/application-policy-infrastructure-controller-apic/200529-Configure-a-Fabric-Extender-with-Applica.html>. For releases preceding Release 3.1(1x), Layer 3 multicast is not supported with FEX. Multicast sources or receivers that are connected to FEX ports are not supported.
- You cannot use a filter with inter-VRF multicast communication.



Note Cisco ACI does not support IP fragmentation. Therefore, when you configure Layer 3 Outside (L3Out) connections to external routers, or Multi-Pod connections through an Inter-Pod Network (IPN), it is recommended that the interface MTU is set appropriately on both ends of a link. On some platforms, such as Cisco ACI, Cisco NX-OS, and Cisco IOS, the configurable MTU value does not take into account the Ethernet headers (matching IP MTU, and excluding the 14-18 Ethernet header size), while other platforms, such as IOS-XR, include the Ethernet header in the configured MTU value. A configured value of 9000 results in a max IP packet size of 9000 bytes in Cisco ACI, Cisco NX-OS, and Cisco IOS, but results in a max IP packet size of 8986 bytes for an IOS-XR untagged interface.

For the appropriate MTU values for each platform, see the relevant configuration guides.

We highly recommend that you test the MTU using CLI-based commands. For example, on the Cisco NX-OS CLI, use a command such as `ping 1.1.1.1 df-bit packet-size 9000 source-interface ethernet 1/1`.

Configuring Layer 3 Multicast Using REST API

Step 1 Configure a tenant and VRF and enable multicast on a VRF.

Example:

```
<fvTenant dn="uni/tn-PIM_Tenant" name="PIM_Tenant">
  <fvCtx knwMcastAct="permit" name="ctx1">
    <pimCtxP mtu="1500">
      </pimCtxP>
    </fvCtx>
  </fvTenant>
```

Step 2 Configure L3 Out and enable multicast (PIM, IGMP) on the L3 Out.

Example:

```
<l3extOut enforceRtctrl="export" name="l3out-pim_l3out1">
  <l3extRsEctx tnFvCtxName="ctx1"/>
  <l3extLNodeP configIssues="" name="bLeaf-CTX1-101">
    <l3extRsNodeL3OutAtt rtrId="200.0.0.1" rtrIdLoopBack="yes" tDn="topology/pod-1/node-101"/>
    <l3extLIIfP name="if-PIM_Tenant-CTX1" tag="yellow-green">
      <igmpIfP/>
      <pimIfP>
```

```

        <pimRsIfPol tDn="uni/tn-PIM_Tenant/pimifpol-pim_poll1"/>
    </pimIfP>
    <l3extRsPathL3OutAtt addr="131.1.1.1/24" ifInstT="l3-port" mode="regular" mtu="1500"
tDn="topology/pod-1/paths-101/pathep-[eth1/46]"/>
    </l3extLIIfP>
    </l3extLNodeP>
    <l3extRsL3DomAtt tDn="uni/l3dom-l3outDom"/>
    <l3extInstP name="l3out-PIM_Tenant-CTX1-ltopo" >
    </l3extInstP>
    <pimExtP enabledAf="ipv4-mcast" name="pim"/>
</l3extOut>

```

Step 3 Configure a BD under the tenant and enable multicast and IGMP on the BD.

Example:

```

<fvTenant dn="uni/tn-PIM_Tenant" name="PIM_Tenant">
  <fvBD arpFlood="yes" mcastAllow="yes" multiDstPktAct="bd-flood" name="bd2" type="regular"
unicastRoute="yes" unkMacUcastAct="flood" unkMcastAct="flood">
    <igmpIfP/>
    <fvRsBDToOut tnL3extOutName="l3out-pim_l3out1"/>
    <fvRsCtx tnFvCtxName="ctx1"/>
    <fvRsIgmprsn/>
    <fvSubnet ctrl="" ip="41.1.1.254/24" preferred="no" scope="private" virtual="no"/>
  </fvBD>
</fvTenant>

```

Step 4 Configure an IGMP policy and assign it to the BD.

Example:

```

<fvTenant dn="uni/tn-PIM_Tenant" name="PIM_Tenant">
  <igmpIfPol grpTimeout="260" lastMbrCnt="2" lastMbrRespTime="1" name="igmp_pol" querierTimeout="255"
queryIntvl="125" robustFac="2" rspIntvl="10" startQueryCnt="2" startQueryIntvl="125" ver="v2">
    </igmpIfPol>
    <fvBD arpFlood="yes" mcastAllow="yes" name="bd2">
      <igmpIfP>
        <igmpRsIfPol tDn="uni/tn-PIM_Tenant/igmpIfPol-igmp_pol"/>
      </igmpIfP>
    </fvBD>
  </fvTenant>

```

Step 5 Configure a route map, PIM, and RP policy on the VRF.

Note When configuring a fabric RP using the REST API, first configure a static RP.

Example:

Configuring a static RP:

```

<fvTenant dn="uni/tn-PIM_Tenant" name="PIM_Tenant">
  <pimRouteMapPol name="rootMap">
    <pimRouteMapEntry action="permit" grp="224.0.0.0/4" order="10" rp="0.0.0.0" src="0.0.0.0/0"/>
  </pimRouteMapPol>
  <fvCtx knwMcastAct="permit" name="ctx1">
    <pimCtxP ctrl="" mtu="1500">
      <pimStaticRPPol>
        <pimStaticRPEntryPol rpIp="131.1.1.2">
          <pimRPGrpRangePol>
            <rtmcRsFilterToRtMapPol tDn="uni/tn-PIM_Tenant/rtmap-rootMap"/>
          </pimRPGrpRangePol>
        </pimStaticRPEntryPol>
      </pimStaticRPPol>
    </pimCtxP>
  </fvCtx>

```

```

    </fvCtx>
  </fvTenant>

```

Configuring a fabric RP:

```

<fvTenant name="t0">
  <pimRouteMapPol name="fabricrp-rtmap">
    <pimRouteMapEntry grp="226.20.0.0/24" order="1" />
  </pimRouteMapPol>
  <fvCtx name="ctx1">
    <pimCtxP ctrl="">
      <pimFabricRPPol status="">
        <pimStaticRPEntryPol rpIp="6.6.6.6">
          <pimRPGrpRangePol>
            <rtdmcRsFilterToRtMapPol tDn="uni/tn-t0/rtmap-fabricrp-rtmap" />
          </pimRPGrpRangePol>
        </pimStaticRPEntryPol>
      </pimFabricRPPol>
    </pimCtxP>
  </fvCtx>
</fvTenant>

```

Step 6 Configure a PIM interface policy and apply it on the L3 Out.

Example:

```

<fvTenant dn="uni/tn-PIM_Tenant" name="PIM_Tenant">
  <pimIfPol authKey="" authT="none" ctrl="" drDelay="60" drPrio="1" helloItvl="30000" itvl="60"
name="pim_poll"/>
  <l3extOut enforceRtctrl="export" name="l3out-pim_l3out1" targetDscp="unspecified">
    <l3extRsEctx tnFvCtxName="ctx1"/>
    <l3extLNodeP name="bLeaf-CTX1-101">
      <l3extRsNodeL3OutAtt rtrId="200.0.0.1" rtrIdLoopBack="yes" tDn="topology/pod-1/node-101"/>
      <l3extLIIfP name="if-SIRI_VPC_src_rcv-CTX1" tag="yellow-green">
        <pimIfP>
          <pimRsIfPol tDn="uni/tn-tn-PIM_Tenant/pimifpol-pim_poll"/>
        </pimIfP>
      </l3extLIIfP>
    </l3extLNodeP>
  </l3extOut>
</fvTenant>

```

Step 7 Configure inter-VRF multicast.

Example:

```

<fvTenant name="t0">
  <pimRouteMapPol name="intervrf" status="">
    <pimRouteMapEntry grp="225.0.0.0/24" order="1" status="" />
    <pimRouteMapEntry grp="226.0.0.0/24" order="2" status="" />
    <pimRouteMapEntry grp="228.0.0.0/24" order="3" status="deleted" />
  </pimRouteMapPol>
  <fvCtx name="ctx1">
    <pimCtxP ctrl="">
      <pimInterVRFPol status="">
        <pimInterVRFEntryPol srcVrfDn="uni/tn-t0/ctx-stig_r_ctx" >
          <rtdmcRsFilterToRtMapPol tDn="uni/tn-t0/rtmap-intervrf" />
        </pimInterVRFEntryPol>
      </pimInterVRFPol>
    </pimCtxP>
  </fvCtx>
</fvTenant>

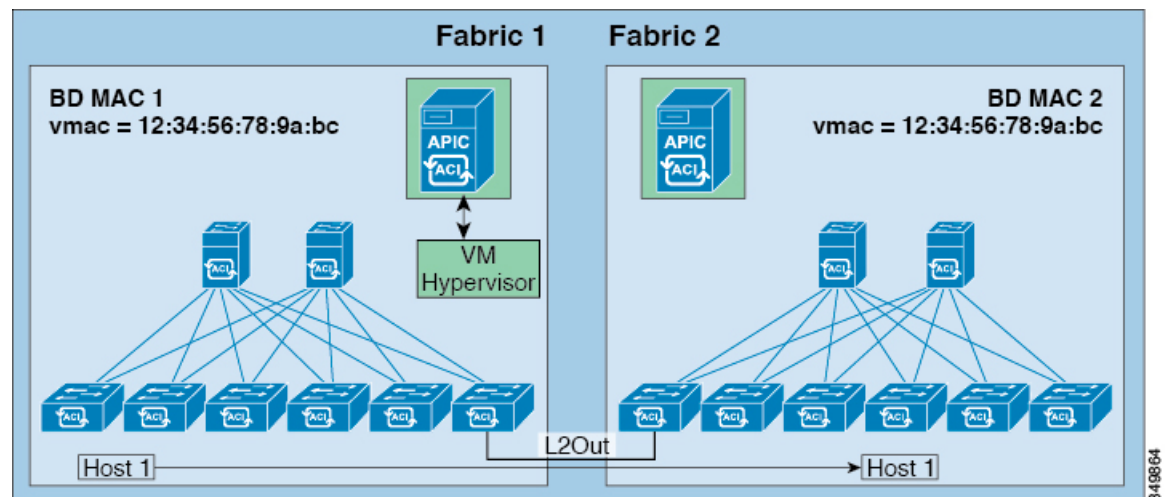
```

Pervasive Gateway

Common Pervasive Gateway

Multiple ACI fabrics can be configured with an IPv4 common gateway on a per bridge domain basis. Doing so enables moving one or more virtual machines (VM) or conventional hosts across the fabrics while the host retains its IP address. VM host moves across fabrics can be done automatically by the VM hypervisor. The ACI fabrics can be co-located, or provisioned across multiple sites. The Layer 2 connection between the ACI fabrics can be a local link, or can be across a routed WAN link. The following figure illustrates the basic common pervasive gateway topology.

Figure 42: ACI Multi-Fabric Common Pervasive Gateway



The per-bridge domain common pervasive gateway configuration requirements are as follows:

- The bridge domain MAC (*mac*) values for each fabric must be unique.



Note The default bridge domain MAC (*mac*) address values are the same for all ACI fabrics. The common pervasive gateway requires an administrator to configure the bridge domain MAC (*mac*) values to be unique for each ACI fabric.

- The bridge domain virtual MAC (*vmac*) address and the subnet virtual IP address must be the same across all ACI fabrics for that bridge domain. Multiple bridge domains can be configured to communicate across connected ACI fabrics. The virtual MAC address and the virtual IP address can be shared across bridge domains.

Configuring Common Pervasive Gateway Using the REST API

Before you begin

- The tenant, VRF, and bridge domain are created.

Configure common pervasive gateway.

In the following example REST API XML, the bolded text is relevant to configuring a common pervasive gateway.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- api/policymgr/mo/.xml -->
<polUni>
  <fvTenant name="test">
    <fvCtx name="test"/>

    <fvBD name="test" vmac="12:34:56:78:9a:bc">
      <fvRsCtx tnFvCtxName="test"/>
      <!-- Primary address -->
      <fvSubnet ip="192.168.15.254/24" preferred="yes"/>
      <!-- Virtual address -->
      <fvSubnet ip="192.168.15.1/24" virtual="yes"/>
    </fvBD>

    <fvAp name="test">
      <fvAEPg name="web">
        <fvRsBd tnFvBDName="test"/>
        <fvRsPathAtt tDn="topology/pod-1/paths-101/pathep-[eth1/3]" encap="vlan-1002"/>
      </fvAEPg>
    </fvAp>
  </fvTenant>
</polUni>
```

Explicit Prefix Lists

About Explicit Prefix List Support for Route Maps/Profile

In Cisco APIC, for public bridge domain (BD) subnets and external transit networks, inbound and outbound route controls are provided through an explicit prefix list. Inbound and outbound route control for Layer 3 Out is managed by the route map/profile (rtctrlProfile). The route map/profile policy supports a fully controllable prefix list for Layer 3 Out in the Cisco ACI fabric.

The subnets in the prefix list can represent the bridge domain public subnets or external networks. Explicit prefix list presents an alternate method and can be used instead of the following:

- Advertising BD subnets through BD to Layer 3 Out relation.

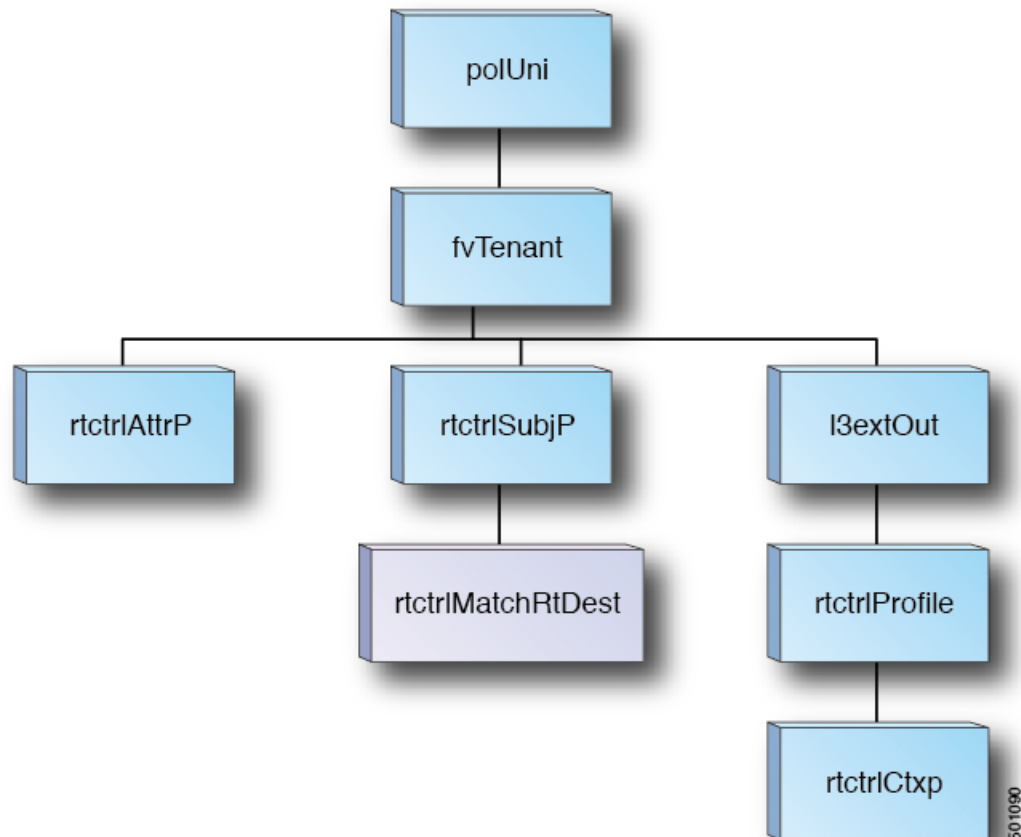


Note The subnet in the BD must be marked public for the subnet to be advertised out.

- Specifying a subnet in the l3extInstP with export/import route control for advertising transit and external networks.

Explicit prefix list is defined through a new match type that is called match route destination (rtctrlMatchRtDest). An example usage is provided in the API example that follows.

Figure 43: External Policy Model of API



Additional information about match rules, set rules when using explicit prefix list are as follows:

Match Rules

- Under the tenant (fvTenant), you can create match profiles (rtctrlSubjP) for route map filtering. Each match profile can contain one or more match rules. Match rule supports multiple match types. Prior to Cisco APIC release 2.1(x), match types supported were explicit prefix list and community list.

Starting with Cisco APIC release 2.1(x), explicit prefix match or match route destination (rtctrlMatchRtDest) is supported.

Match prefix list (rtctrlMatchRtDest) supports one or more subnets with an optional aggregate flag. Aggregate flags are used for allowing prefix matches with multiple masks starting with the mask mentioned in the configuration till the maximum mask allowed for the address family of the prefix. This is the equivalent of the "le" option in the prefix-list in NX-OS software (example, 10.0.0.0/8 le 32).

The prefix list can be used for covering the following cases:

- Allow all (0.0.0.0/0 with aggregate flag, equivalent of 0.0.0.0/0 le 32)
- One or more of specific prefixes (example: 10.1.1.0/24)
- One or more of prefixes with aggregate flag (example, equivalent of 10.1.1.0/24 le 32).



Note When a route map with a match prefix “0.0.0.0/0 with aggregate flag” is used under an L3Out EPG in the export direction, the rule is applied only for redistribution from dynamic routing protocols. Therefore, the rule is not applied to the following (in routing protocol such as OSPF or EIGRP):

- Bridge domain (BD) subnets
- Directly connected subnets on the border leaf switch
- Static routes defined on the L3Out

- The explicit prefix match rules can contain one or more subnets, and these subnets can be bridge domain public subnets or external networks. Subnets can also be aggregated up to the maximum subnet mask (/32 for IPv4 and /128 for IPv6).
- When multiple match rules of different types are present (such as match community and explicit prefix match), the match rule is allowed only when the match statements of all individual match types match. This is the equivalent of the AND filter. The explicit prefix match is contained by the subject profile (rtctrlSubjP) and will form a logical AND if other match rules are present under the subject profile.
- Within a given match type (such as match prefix list), at least one of the match rules statement must match. Multiple explicit prefix match (rtctrlMatchRtDest) can be defined under the same subject profile (rtctrlSubjP) which will form a logical OR.

Set Rules

- Set policies must be created to define set rules that are carried with the explicit prefixes such as set community, set tag.

Guidelines and Limitations

- You must choose one of the following two methods to configure your route maps. If you use both methods, it will result in double entries and undefined route maps.
 - Add routes under the bridge domain (BD) and configure a BD to Layer 3 Outside relation
 - Configure the match prefix under rtctrlSubjP match profiles.

- Starting 2.3(x), **deny-static** implicit entry has been removed from Export Route Map. The user needs to configure explicitly the permit and deny entries required to control the export of static routes.
- Route-map per peer in an L3Out is not supported. Route-map can only be applied on L3Out as a whole.

Following are possible workarounds to this issue:

- Block the prefix from being advertised from the other side of the neighbor.
- Block the prefix on the route-map on the existing L3Out where you don't want to learn the prefix, and move the neighbor to another L3Out where you want to learn the prefix and create a separate route-map.
- Creating route-maps using a mixture of GUI and API commands is not supported. As a possible workaround, you can create a route-map different from the default route-map using the GUI, but the route-map created through the GUI on an L3Out cannot be applied to per-peer.

About Route Map/Profile

The route profile is a logical policy that defines an ordered set (rtctrlCtxP) of logical match action rules with associated set action rules. The route profile is the logical abstract of a route map. Multiple route profiles can be merged into a single route map. A route profile can be one of the following types:

- Match Prefix and Routing Policy: Pervasive subnets (fvSubnet) and external subnets (l3extSubnet) are combined with a route profile and merged into a single route map (or route map entry). Match Prefix and Routing Policy is the default value.
- Match Routing Policy Only: The route profile is the only source of information to generate a route map, and it will overwrite other policy attributes.



Note When explicit prefix list is used, the type of the route profile should be set to "match routing policy only".

After the match and set profiles are defined, the route map must be created in the Layer 3 Out. Route maps can be created using one of the following methods:

- Create a "default-export" route map for export route control, and a "default-import" route map for import route control.
- Create other route maps (not named default-export or default-import) and setup the relation from one or more l3extInstPs or subnets under the l3extInstP.
- In either case, match the route map on explicit prefix list by pointing to the rtctrlSubjP within the route map.

In the export and import route map, the set and match rules are grouped together along with the relative sequence across the groups (rtctrlCtxP). Additionally, under each group of match and set statements (rtctrlCtxP) the relation to one or more match profiles are available (rtctrlSubjP).

Any protocol enabled on Layer 3 Out (for example BGP protocol), will use the export and import route map for route filtering.

Aggregation Support for Explicit Prefix List

Each prefix (rtctrlMatchRtDest) in the match prefixes list can be aggregated to support multiple subnets matching with one prefix list entry.

Aggregated Prefixes and BD Private Subnets

Although subnets in the explicit prefix list match may match the BD private subnets using aggregated or exact match, private subnets will not be advertised through the routing protocol using the explicit prefix list. The scope of the BD subnet must be set to "public" for the explicit prefix list feature to advertise the BD subnets.

Differences in Behavior for 0.0.0.0/0 with Aggregation

The 0.0.0.0/0 with Aggregate configuration creates an IP prefix-list equivalent to "0.0.0.0/0 le 32". The 0.0.0.0/0 with Aggregate configuration can be used mainly in two situations:

- "Export Route Control Subnet" with "Aggregate Export" scope in L3Out subnet under the L3Out network (L3Out EPG)
- An explicit prefix-list (Match Prefix rule) assigned to a route map with the name "default-export"

When used with the "Export Route Control Subnet" scope under the L3Out subnet, the route map will only match routes learned from dynamic routing protocols. It will not match BD subnets or directly-connected networks.

When used with the explicit route map configuration, the route map will match all routes, including BD subnets and directly-connected networks.

Consider the following examples to get a better understanding of the expected and unexpected (inconsistent) behavior in the two situations described above.

Scenario 1

For the first scenario, we configure a route map (with a name of rpm_with_catch_all) using a configuration post similar to the following:

```
<l3extOut annotation="" descr="" dn="uni/tn-t9/out-L3-out" enforceRtctrl="export"
name="L3-out" nameAlias="" ownerKey="" ownerTag="" targetDscp="unspecified">
  <rtctrlProfile annotation="" descr="" name="rpm_with_catch_all" nameAlias="" ownerKey=""
ownerTag="" type="combinable">
    <rtctrlCtxP action="permit" annotation="" descr="" name="catch_all" nameAlias=""
order="0">
      <rtctrlScope annotation="" descr="" name="" nameAlias="">
        <rtctrlRsScopeToAttrP annotation="" tnRtctrlAttrPName="set_metric_type"/>
      </rtctrlScope>
    </rtctrlCtxP>
  </rtctrlProfile>
  <ospfExtP annotation="" areaCost="1" areaCtrl="redistribute,summary" areaId="backbone"
areaType="regular" descr="" multipodInternal="no" nameAlias=""/>
  <l3extRsEctx annotation="" tnFvCtxName="ctx0"/>
  <l3extLNodeP annotation="" configIssues="" descr="" name="leaf" nameAlias="" ownerKey=""
ownerTag="" tag="yellow-green" targetDscp="unspecified">
    <l3extRsNodeL3OutAtt annotation="" configIssues="" rtrId="20.2.0.2" rtrIdLoopBack="no"
tDn="topology/pod-1/node-104">
      <l3extLoopBackIfP addr="14.1.1.1/32" annotation="" descr="" name="" nameAlias=""/>
    </l3extRsNodeL3OutAtt>
  <l3extInfraNodeP annotation="" descr="" fabricExtCtrlPeering="no"
fabricExtIntersiteCtrlPeering="no" name="" nameAlias="" spineRole=""/>
</l3extRsNodeL3OutAtt>
```

```

        <l3extLIfP annotation="" descr="" name="interface" nameAlias="" ownerKey=""
ownerTag="" tag="yellow-green">
        <ospfIfP annotation="" authKeyId="1" authType="none" descr="" name=""
nameAlias="">
            <ospfRsIfPol annotation="" tnOspfIfPolName=""/>
        </ospfIfP>
        <l3extRsPathL3OutAtt addr="36.1.1.1/24" annotation="" autostate="disabled"
descr="" encap="vlan-3063" encapScope="local" ifInstT="ext-svi" ipv6Dad="enabled" llAddr="::"
mac="00:22:BD:F8:19:FF" mode="regular" mtu="inherit"
tDn="topology/pod-1/paths-104/pathep-[accBndlGrp_104_pc13]" targetDscp="unspecified"/>
        <l3extRsNdIfPol annotation="" tnNdIfPolName=""/>
        <l3extRsIngressQosDppPol annotation="" tnQosDppPolName=""/>
        <l3extRsEgressQosDppPol annotation="" tnQosDppPolName=""/>
    </l3extLIfP>
</l3extLNodeP>
    <l3extInstP annotation="" descr="" exceptionTag="" floodOnEncap="disabled"
matchT="AtleastOne" name="epg" nameAlias="" prefGrMemb="exclude" prio="unspecified"
targetDscp="unspecified">
        <l3extRsInstPToProfile annotation="" direction="export"
tnRtctrlProfileName="rpm_with_catch_all"/>
        <l3extSubnet aggregate="" annotation="" descr="" ip="0.0.0.0/0" name="" nameAlias=""
scope="import-security"/>
        <fvRsCustQosPol annotation="" tnQosCustomPolName=""/>
    </l3extInstP>
</l3extOut>

<rtctrlAttrP annotation="" descr="" dn="uni/tn-t9/attr-set_metric_type" name="set_metric_type"
nameAlias="">
    <rtctrlSetRtMetricType annotation="" descr="" metricType="ospf-type1" name="" nameAlias=""
type="metric-type"/>
</rtctrlAttrP>

<rtctrlSubjP annotation="" descr="" dn="uni/tn-t9/subj-catch_all_ip" name="catch_all_ip"
nameAlias="">
    <rtctrlMatchRtDest aggregate="yes" annotation="" descr="" ip="0.0.0.0/0" name=""
nameAlias=""/>
</rtctrlSubjP>

```

With this route map, what we would expect with 0.0.0.0/0 is that all the routes would go with the property `metricType="ospf-type1"`, but only for the OSPF route.

In addition, we also have a subnet configured under a bridge domain (for example, 209.165.201.0/27), with a bridge domain to L3Out relation, using a route map with a pervasive subnet (fvSubnet) for a static route. However, even though the route map shown above is combinable, we do not want it applied for the subnet configured under the bridge domain, because we want 0.0.0.0/0 in the route map above to apply only for the transit route, not on the static route.

Following is the output for the `show route-map` and `show ip prefix-list` commands, where `exp-ctx-st-2555939` is the name of the outbound route map for the subnet configured under the bridge domain, and the name of the prefix list is provided within the output from the `show route-map` command:

```

leaf4# show route-map exp-ctx-st-2555939
route-map exp-ctx-st-2555939, deny, sequence 1
  Match clauses:
    tag: 4294967295
  Set clauses:
route-map exp-ctx-st-2555939, permit, sequence 15801
  Match clauses:
    ip address prefix-lists: IPv4-st16391-2555939-exc-int-inferred-export-dst
    ipv6 address prefix-lists: IPv6-deny-all

```

Set clauses:

```
leaf4# show ip prefix-list IPv4-st16391-2555939-exc-int-inferred-export-dst
ip prefix-list IPv4-st16391-2555939-exc-int-inferred-export-dst: 1 entries
seq 1 permit 209.165.201.0/27
```

```
leaf4#
```

In this situation, everything behaves as expected, because when the bridge domain subnet goes out, it is not applying the `rpm_with_catch_all` route map policies.

Scenario 2

For the second scenario, we configure a "default-export" route map for export route control, where an explicit prefix-list (Match Prefix rule) is assigned to the "default-export" route map, using a configuration post similar to the following:

```
<l3extOut annotation="" descr="" dn="uni/tn-t9/out-L3-out" enforceRtctrl="export"
name="L3-out" nameAlias="" ownerKey="" ownerTag="" targetDscp="unspecified">
  <rtctrlProfile annotation="" descr="" name="default-export" nameAlias="" ownerKey=""
ownerTag="" type="combinable">
    <rtctrlCtxP action="permit" annotation="" descr="" name="set-rule" nameAlias=""
order="0">
      <rtctrlScope annotation="" descr="" name="" nameAlias="">
        <rtctrlRsScopeToAttrP annotation="" tnRtctrlAttrPName="set_metric_type"/>
      </rtctrlScope>
    </rtctrlCtxP>
  </rtctrlProfile>
  <ospfExtP annotation="" areaCost="1" areaCtrl="redistribute,summary" areaId="backbone"
areaType="regular" descr="" multipodInternal="no" nameAlias=""/>
  <l3extRsEctx annotation="" tnFvCtxName="ctx0"/>
  <l3extLNodeP annotation="" configIssues="" descr="" name="leaf" nameAlias="" ownerKey=""
ownerTag="" tag="yellow-green" targetDscp="unspecified">
    <l3extRsNodeL3OutAtt annotation="" configIssues="" rtrId="20.2.0.2" rtrIdLoopBack="no"
tDn="topology/pod-1/node-104">
      <l3extLoopBackIfP addr="14.1.1.1/32" annotation="" descr="" name="" nameAlias=""/>
      <l3extInfraNodeP annotation="" descr="" fabricExtCtrlPeering="no"
fabricExtIntersiteCtrlPeering="no" name="" nameAlias="" spineRole=""/>
      </l3extRsNodeL3OutAtt>
      <l3extLIIfP annotation="" descr="" name="interface" nameAlias="" ownerKey=""
ownerTag="" tag="yellow-green">
        <ospfIfP annotation="" authKeyId="1" authType="none" descr="" name=""
nameAlias="">
          <ospfRsIfPol annotation="" tnOspfIfPolName=""/>
        </ospfIfP>
        <l3extRsPathL3OutAtt addr="36.1.1.1/24" annotation="" autostate="disabled"
descr="" encap="vlan-3063" encapScope="local" ifInstT="ext-svi" ipv6Dad="enabled" llAddr=":."
mac="00:22:BD:F8:19:FF" mode="regular" mtu="inherit"
tDn="topology/pod-1/paths-104/pathep-[accBndlGrp_104_pc13]" targetDscp="unspecified"/>
          <l3extRsNdIfPol annotation="" tnNdIfPolName=""/>
          <l3extRsIngressQosDppPol annotation="" tnQosDppPolName=""/>
          <l3extRsEgressQosDppPol annotation="" tnQosDppPolName=""/>
        </l3extLIIfP>
      </l3extLNodeP>
      <l3extInstP annotation="" descr="" exceptionTag="" floodOnEncap="disabled"
matchT="AtleastOne" name="epg" nameAlias="" prefGrMemb="exclude" prio="unspecified"
targetDscp="unspecified">
        <l3extSubnet aggregate="" annotation="" descr="" ip="0.0.0.0/0" name="" nameAlias=""
scope="import-security"/>
        <fvRsCustQosPol annotation="" tnQosCustomPolName=""/>
      </l3extInstP>
    </l3extOut>
```

Notice that this `default-export` route map has similar information as the `rpm_with_catch_all` route map, where the IP is set to `0.0.0.0/0` (`ip=0.0.0.0/0`), and the set rule in the `default-export` route map is configured only with the Set Metric Type (`tnRtctrlAttrPName=set_metric_type`).

Similar to the situation in the previous example, we also have the same subnet configured under the bridge domain, with a bridge domain to L3Out relation, as we did in the previous example.

However, following is the output in this scenario for the `show route-map` and `show ip prefix-list` commands:

```
leaf4# show route-map exp-ctx-st-2555939
route-map exp-ctx-st-2555939, deny, sequence 1
  Match clauses:
    tag: 4294967295
  Set clauses:
route-map exp-ctx-st-2555939, permit, sequence 8201
  Match clauses:
    ip address prefix-lists:
IPv4-st16391-2555939-exc-int-out-default-export2set-rule0pfx-only-dst
    ipv6 address prefix-lists: IPv6-deny-all
  Set clauses:
metric-type type-1

leaf4# show ip prefix-list IPv4-st16391-2555939-exc-int-inferred-export-dst
% Policy IPv4-st16391-2555939-exc-int-inferred-export-dst not found
ifav82-leaf4# show ip prefix-list
IPv4-st16391-2555939-exc-int-out-default-export2set-rule0pfx-only-dst
ip prefix-list IPv4-st16391-2555939-exc-int-out-default-export2set-rule0pfx-only-dst: 1
entries
  seq 1 permit 209.165.201.0/27

leaf4#
```

Notice that in this situation, when the bridge domain subnet goes out, it is applying the `default-export` route map policies. In this situation, that route map matches all routes, including BD subnets and directly-connected networks. This is inconsistent behavior.

Configuring Route Map/Profile with Explicit Prefix List Using REST API

Before you begin

- Tenant and VRF must be configured.

Configure the route map/profile using explicit prefix list.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<fvTenant name="PM" status="">
  <rtctrlAttrP name="set_dest">
    <rtctrlSetComm community="regular:as2-nn2:5:24" />
  </rtctrlAttrP>
  <rtctrlSubjP name="allow_dest">
    <rtctrlMatchRtDest ip="192.169.0.0/24" />
    <rtctrlMatchCommTerm name="term1">
      <rtctrlMatchCommFactor community="regular:as2-nn2:5:24" status="" />
      <rtctrlMatchCommFactor community="regular:as2-nn2:5:25" status="" />
    </rtctrlMatchCommTerm>
    <rtctrlMatchCommRegexTerm commType="regular" regex="200:*" status="" />
  </rtctrlSubjP>
</fvTenant>
```

```

</rtctrlSubjP>
<rtctrlSubjP name="deny_dest">
  <rtctrlMatchRtDest ip="192.168.0.0/24" />
</rtctrlSubjP>
<fvCtx name="ctx" />
<l3extOut name="L3Out_1" enforceRtctrl="import,export" status="">
  <l3extRsEctx tnFvCtxName="ctx" />
  <l3extLNodeP name="bLeaf">
    <l3extRsNodeL3OutAtt tDn="topology/pod-1/node-101" rtrId="1.2.3.4" />
    <l3extLIIfP name="portIf">
      <l3extRsPathL3OutAtt tDn="topology/pod-1/paths-101/pathep-[eth1/25]" ifInstT="sub-interface"
encap="vlan-1503" addr="10.11.12.11/24" />
      <ospfIfP />
    </l3extLIIfP>
    <bgpPeerP addr="5.16.57.18/32" ctrl="send-com" />
    <bgpPeerP addr="6.16.57.18/32" ctrl="send-com" />
  </l3extLNodeP>
<bgpExtP />
<ospfExtP areaId="0.0.0.59" areaType="nssa" status="" />
<l3extInstP name="l3extInstP_1" status="">
  <l3extSubnet ip="17.11.1.11/24" scope="import-security" />
</l3extInstP>
<rtctrlProfile name="default-export" type="global" status="">
  <rtctrlCtxP name="ctx_deny" action="deny" order="1">
    <rtctrlRsCtxPToSubjP tnRtctrlSubjPName="deny_dest" status="" />
  </rtctrlCtxP>
  <rtctrlCtxP name="ctx_allow" order="2">
    <rtctrlRsCtxPToSubjP tnRtctrlSubjPName="allow_dest" status="" />
  </rtctrlCtxP>
  <rtctrlScope name="scope" status="">
    <rtctrlRsScopeToAttrP tnRtctrlAttrPName="set_dest" status="" />
  </rtctrlScope>
</rtctrlProfile>
</l3extOut>
<fvBD name="testBD">
  <fvRsBDToOut tnL3extOutName="L3Out_1" />
  <fvRsCtx tnFvCtxName="ctx" />
  <fvSubnet ip="40.1.1.12/24" scope="public" />
  <fvSubnet ip="40.1.1.2/24" scope="private" />
  <fvSubnet ip="2003::4/64" scope="public" />
</fvBD>
</fvTenant>

```

IP Address Aging Tracking

Overview

The IP Aging policy tracks and ages unused IP addresses on an endpoint. Tracking is performed using the endpoint retention policy configured for the bridge domain to send ARP requests (for IPv4) and neighbor solicitations (for IPv6) at 75% of the local endpoint aging interval. When no response is received from an IP address, that IP address is aged out.

This document explains how to configure the IP Aging policy.

Configuring IP Aging Using the REST API

This section explains how to enable and disable the IP aging policy using the REST API.

Step 1 To enable the IP aging policy:

Example:

```
<epIpAgingP adminSt="enabled" descr="" dn="uni/infra/ipAgingP-default" name="default" ownerKey=""
ownerTag=""/>
```

Step 2 To disable the IP aging policy:

Example:

```
<epIpAgingP adminSt="disabled" descr="" dn="uni/infra/ipAgingP-default" name="default" ownerKey=""
ownerTag=""/>
```

What to do next

To specify the interval used for tracking IP addresses on endpoints, create an Endpoint Retention policy by sending a post with XML such as the following example:

```
<fvEpRetPol bounceAgeIntvl="630" bounceTrig="protocol"
holdIntvl="350" lcOwn="local" localEpAgeIntvl="900" moveFreq="256"
name="EndpointPol1" remoteEpAgeIntvl="350"/>
```

Route Summarization

Configuring Route Summarization for BGP, OSPF, and EIGRP Using the REST API

Step 1 Configure BGP route summarization using the REST API as follows:

Example:

```
<fvTenant name="common">
  <fvCtx name="vrf1"/>
  <bgpRtSummPol name="bgp_rt_summ" cntrl='as-set'/>
  <l3extOut name="l3_ext_pol" >
    <l3extLNodeP name="bLeaf">
      <l3extRsNodeL3OutAtt tDn="topology/pod-1/node-101" rtrId="20.10.1.1"/>
      <l3extLIIfP name='portIf'>
        <l3extRsPathL3OutAtt tDn="topology/pod-1/paths-101/pathep-[eth1/31]" ifInstT='l3-port'
addr="10.20.1.3/24"/>
      </l3extLIIfP>
    </l3extLNodeP>
  </bgpExtP />
  <l3extInstP name="InstP" >
  <l3extSubnet ip="10.0.0.0/8" scope="export-rtctrl">
    <l3extRsSubnetToRtSumm tDn="uni/tn-common/bgprtsum-bgp_rt_summ"/>
    <l3extRsSubnetToProfile tnRtctrlProfileName="rtprof"/>
  </l3extRsSubnetToProfile />
  </l3extSubnet />
  </l3extInstP />
</fvTenant>
```

```

        </l3extSubnet>
    </l3extInstP>
    <l3extRsEctx tnFvCtxName="vrf1"/>
</l3extOut>
</fvTenant>

```

Step 2 Configure OSPF inter-area and external summarization using the following REST API:

Example:

```

<?xml version="1.0" encoding="utf-8"?>
<fvTenant name="t20">
  <!--Ospf Inter External route summarization Policy-->
  <ospfRtSummPol cost="unspecified" interAreaEnabled="no" name="ospfext"/>
  <!--Ospf Inter Area route summarization Policy-->
  <ospfRtSummPol cost="16777215" interAreaEnabled="yes" name="interArea"/>
  <fvCtx name="ctx0" pcEnfDir="ingress" pcEnfPref="enforced"/>
  <!-- L3OUT backbone Area-->
  <l3extOut enforceRtctrl="export" name="l3_1" ownerKey="" ownerTag="" targetDscp="unspecified">
    <l3extRsEctx tnFvCtxName="ctx0"/>
    <l3extLNodeP name="node-101">
      <l3extRsNodeL3OutAtt rtrId="20.1.3.2" rtrIdLoopBack="no" tDn="topology/pod-1/node-101"/>
      <l3extLIIfP name="intf-1">
        <l3extRsPathL3OutAtt addr="20.1.5.2/24" encap="vlan-1001" ifInstT="sub-interface"
tDn="topology/pod-1/paths-101/pathep-[eth1/33]"/>
      </l3extLIIfP>
    </l3extLNodeP>
    <l3extInstP name="l3InstP1">
      <fvRsProv tnVzBrCPName="default"/>
      <!--Ospf External Area route summarization-->
      <l3extSubnet aggregate="" ip="193.0.0.0/8" name="" scope="export-rtctrl">
        <l3extRsSubnetToRtSumm tDn="uni/tn-t20/ospfrtsumm-ospfext"/>
      </l3extSubnet>
    </l3extInstP>
    <ospfExtP areaCost="1" areaCtrl="redistribute,summary" areaId="backbone" areaType="regular"/>
  </l3extOut>
  <!-- L3OUT Regular Area-->
  <l3extOut enforceRtctrl="export" name="l3_2">
    <l3extRsEctx tnFvCtxName="ctx0"/>
    <l3extLNodeP name="node-101">
      <l3extRsNodeL3OutAtt rtrId="20.1.3.2" rtrIdLoopBack="no" tDn="topology/pod-1/node-101"/>
      <l3extLIIfP name="intf-2">
        <l3extRsPathL3OutAtt addr="20.1.2.2/24" encap="vlan-1014" ifInstT="sub-interface"
tDn="topology/pod-1/paths-101/pathep-[eth1/11]"/>
      </l3extLIIfP>
    </l3extLNodeP>
    <l3extInstP matchT="AtleastOne" name="l3InstP2">
      <fvRsCons tnVzBrCPName="default"/>
      <!--Ospf Inter Area route summarization-->
      <l3extSubnet aggregate="" ip="197.0.0.0/8" name="" scope="export-rtctrl">
        <l3extRsSubnetToRtSumm tDn="uni/tn-t20/ospfrtsumm-interArea"/>
      </l3extSubnet>
    </l3extInstP>
    <ospfExtP areaCost="1" areaCtrl="redistribute,summary" areaId="0.0.0.57" areaType="regular"/>
  </l3extOut>
</fvTenant>

```

Step 3 Configure EIGRP summarization using the following REST API:

Example:

```
<fvTenant name="exampleCorp">
  <l3extOut name="out1">
    <l3extInstP name="eigrpSummInstp" >
      <l3extSubnet aggregate="" descr="" ip="197.0.0.0/8" name="" scope="export-rtctrl">
        <l3extRsSubnetToRtSumm/>
      </l3extSubnet>
    </l3extInstP>
  </l3extOut>
  <eigrpRtSummPol name="poll" />
</fvTenant>
```

Note There is no route summarization policy to be configured for EIGRP. The only configuration needed for enabling EIGRP summarization is the summary subnet under the InstP.

Configuring Route Summarization for BGP, OSPF, and EIGRP Using the REST API

Step 1 Configure BGP route summarization using the REST API as follows:

Example:

```
<fvTenant name="common">
  <fvCtx name="vrf1"/>
  <bgpRtSummPol name="bgp_rt_summ" cntrl='as-set' />
  <l3extOut name="l3_ext_pol" >
    <l3extLNodeP name="bLeaf">
      <l3extRsNodeL3OutAtt tDn="topology/pod-1/node-101" rtrId="20.10.1.1"/>
      <l3extLIIfP name='portIf'>
        <l3extRsPathL3OutAtt tDn="topology/pod-1/paths-101/pathep-[eth1/31]" ifInstT='l3-port'
addr="10.20.1.3/24"/>
      </l3extLIIfP>
    </l3extLNodeP>
  <bgpExtP />
  <l3extInstP name="InstP" >
    <l3extSubnet ip="10.0.0.0/8" scope="export-rtctrl">
      <l3extRsSubnetToRtSumm tDn="uni/tn-common/bgprtsum-bgp_rt_summ"/>
      <l3extRsSubnetToProfile tnRtctrlProfileName="rtprof"/>
    </l3extSubnet>
  </l3extInstP>
  <l3extRsEctx tnFvCtxName="vrf1"/>
</l3extOut>
</fvTenant>
```

Step 2 Configure OSPF inter-area and external summarization using the following REST API:

Example:

```
<?xml version="1.0" encoding="utf-8"?>
<fvTenant name="t20">
  <!--Ospf Inter External route summarization Policy-->
  <ospfRtSummPol cost="unspecified" interAreaEnabled="no" name="ospfext"/>
  <!--Ospf Inter Area route summarization Policy-->
  <ospfRtSummPol cost="16777215" interAreaEnabled="yes" name="interArea"/>
  <fvCtx name="ctx0" pcEnfDir="ingress" pcEnfPref="enforced"/>
  <!-- L3OUT backbone Area-->
```

```

<l3extOut enforceRtctrl="export" name="l3_1" ownerKey="" ownerTag="" targetDscp="unspecified">
  <l3extRsEctx tnFvCtxName="ctx0"/>
  <l3extLNodeP name="node-101">
    <l3extRsNodeL3OutAtt rtrId="20.1.3.2" rtrIdLoopBack="no" tDn="topology/pod-1/node-101"/>
    <l3extLIIfP name="intf-1">
      <l3extRsPathL3OutAtt addr="20.1.5.2/24" encap="vlan-1001" ifInstT="sub-interface"
tDn="topology/pod-1/paths-101/pathsep-[eth1/33]"/>
    </l3extLIIfP>
  </l3extLNodeP>
  <l3extInstP name="l3InstP1">
    <fvRsProv tnVzBrCPName="default"/>
    <!--Ospf External Area route summarization-->
    <l3extSubnet aggregate="" ip="193.0.0.0/8" name="" scope="export-rtctrl">
      <l3extRsSubnetToRtSumm tDn="uni/tn-t20/ospfrtsumm-ospfext"/>
    </l3extSubnet>
  </l3extInstP>
  <ospfExtP areaCost="1" areaCtrl="redistribute,summary" areaId="backbone" areaType="regular"/>
</l3extOut>
<!-- L3OUT Regular Area-->
<l3extOut enforceRtctrl="export" name="l3_2">
  <l3extRsEctx tnFvCtxName="ctx0"/>
  <l3extLNodeP name="node-101">
    <l3extRsNodeL3OutAtt rtrId="20.1.3.2" rtrIdLoopBack="no" tDn="topology/pod-1/node-101"/>
    <l3extLIIfP name="intf-2">
      <l3extRsPathL3OutAtt addr="20.1.2.2/24" encap="vlan-1014" ifInstT="sub-interface"
tDn="topology/pod-1/paths-101/pathsep-[eth1/11]"/>
    </l3extLIIfP>
  </l3extLNodeP>
  <l3extInstP matchT="AtleastOne" name="l3InstP2">
    <fvRsCons tnVzBrCPName="default"/>
    <!--Ospf Inter Area route summarization-->
    <l3extSubnet aggregate="" ip="197.0.0.0/8" name="" scope="export-rtctrl">
      <l3extRsSubnetToRtSumm tDn="uni/tn-t20/ospfrtsumm-interArea"/>
    </l3extSubnet>
  </l3extInstP>
  <ospfExtP areaCost="1" areaCtrl="redistribute,summary" areaId="0.0.0.57" areaType="regular"/>
</l3extOut>
</fvTenant>

```

Step 3 Configure EIGRP summarization using the following REST API:

Example:

```

<fvTenant name="exampleCorp">
  <l3extOut name="out1">
    <l3extInstP name="eigrpSummInstp" >
      <l3extSubnet aggregate="" descr="" ip="197.0.0.0/8" name="" scope="export-rtctrl">
        <l3extRsSubnetToRtSumm/>
      </l3extSubnet>
    </l3extInstP>
  </l3extOut>
  <eigrpRtSummPol name="poll" />

```

Note There is no route summarization policy to be configured for EIGRP. The only configuration needed for enabling EIGRP summarization is the summary subnet under the InstP.

Route Controls

About Configuring a Routing Control Protocol Using Import and Export Controls

This topic provides a typical example that shows how to configure a routing control protocol using import and export controls. It assumes that you have configured Layer 3 outside network connections with BGP. You can also perform these tasks for a Layer 3 outside network configured with OSPF.



Note Cisco ACI does not support IP fragmentation. Therefore, when you configure Layer 3 Outside (L3Out) connections to external routers, or Multi-Pod connections through an Inter-Pod Network (IPN), it is recommended that the interface MTU is set appropriately on both ends of a link. On some platforms, such as Cisco ACI, Cisco NX-OS, and Cisco IOS, the configurable MTU value does not take into account the Ethernet headers (matching IP MTU, and excluding the 14-18 Ethernet header size), while other platforms, such as IOS-XR, include the Ethernet header in the configured MTU value. A configured value of 9000 results in a max IP packet size of 9000 bytes in Cisco ACI, Cisco NX-OS, and Cisco IOS, but results in a max IP packet size of 8986 bytes for an IOS-XR untagged interface.

For the appropriate MTU values for each platform, see the relevant configuration guides.

We highly recommend that you test the MTU using CLI-based commands. For example, on the Cisco NX-OS CLI, use a command such as `ping 1.1.1.1 df-bit packet-size 9000 source-interface ethernet 1/1`.

Configuring a Route Control Protocol to Use Import and Export Controls, With the REST API

This example assumes that you have configured the Layer 3 outside network connections using BGP. It is also possible to perform these tasks for a network using OSPF.

Before you begin

- The tenant, private network, and bridge domain are created.
- The Layer 3 outside tenant network is configured.

Configure the route control protocol using import and export controls.

Example:

```
<l3extOut descr="" dn="/tn-Ten_ND/out-L3Out1" enforceRtctrl="export" name="L3Out1" ownerKey=""
ownerTag="" targetDscp="unspecified">
  <l3extLNodeP descr="" name="LNodeP1" ownerKey="" ownerTag="" tag="yellow-green"
targetDscp="unspecified">
    <l3extRsNodeL3OutAtt rtrId="1.2.3.4" rtrIdLoopBack="yes" tDn="/topology/pod-1/node-101">
      <l3extLoopBackIfP addr="2000::3" descr="" name=""/>
    </l3extRsNodeL3OutAtt>
    <l3extLIIfP descr="" name="IFP1" ownerKey="" ownerTag="" tag="yellow-green">
      <ospfIfP authKeyId="1" authType="none" descr="" name="">
```

```

        <ospfRsIfPol tnOspfIfPolName=""/>
    </ospfIfP>
    <l3extRsNdIfPol tnNdIfPolName=""/>
    <l3extRsPathL3OutAtt addr="10.11.12.10/24" descr="" encap="unknown" ifInstT="l3-port"
llAddr="::" mac="00:22:BD:F8:19:FF" mtu="1500" tDn="topology/pod-1/paths-101/pathep-[eth1/17]"
targetDscp="unspecified"/>
    </l3extLIIfP>
</l3extLNodeP>
<l3extRsEctx tnFvCtxName="PVN1"/>
<l3extInstP descr="" matchT="AtleastOne" name="InstP1" prio="unspecified"
targetDscp="unspecified">
    <fvRsCustQosPol tnQosCustomPolName=""/>
    <l3extSubnet aggregate="" descr="" ip="192.168.1.0/24" name="" scope=""/>
</l3extInstP>
<ospfExtP areaCost="1" areaCtrl="redistribute,summary" areaId="0.0.0.1" areaType="nssa"
descr=""/>
<rtctrlProfile descr="" name="default-export" ownerKey="" ownerTag="">
    <rtctrlCtxP descr="" name="routecontrolpvtnw" order="3">
        <rtctrlScope descr="" name="">
            <rtctrlRsScopeToAttrP tnRtctrlAttrPName="actionruleprofile2"/>
        </rtctrlScope>
    </rtctrlCtxP>
</rtctrlProfile>
</l3extOut>

```

Layer 3 to Layer 3 Out Inter-VRF Leaking

Layer 3 Out to Layer 3 Out Inter-VRF Leaking

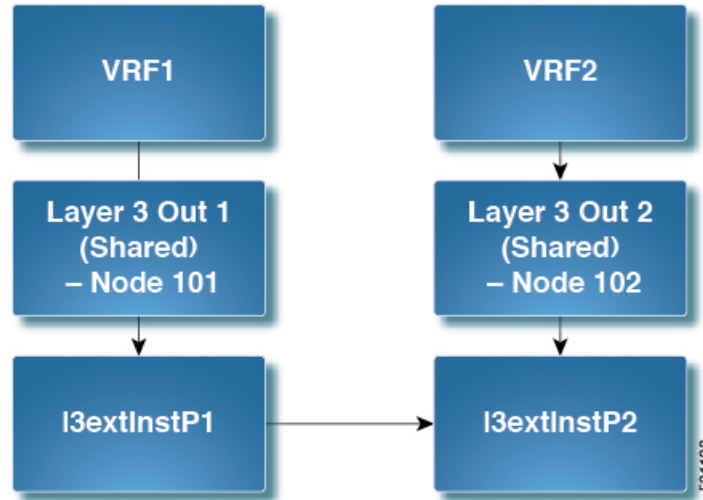
Starting with Cisco APIC release 2.2(2e), when there are two Layer 3 Outs in two different VRFs, inter-VRF leaking is supported.

For this feature to work, the following conditions must be satisfied:

- A contract between the two Layer 3 Outs is required.
- Routes of connected and transit subnets for a Layer 3 Out are leaked by enforcing contracts (L3Out-L3Out as well as L3Out-EPG) and without leaking the dynamic or static routes between VRFs.
- Dynamic or static routes are leaked for a Layer 3 Out by enforcing contracts (L3Out-L3Out as well as L3Out-EPG) and without advertising directly connected or transit routes between VRFs.
- Shared Layer 3 Outs in different VRFs can communicate with each other.
- There is no associated L3Out required for the bridge domain. When an Inter-VRF shared L3Out is used, it is not necessary to associate the user tenant bridge domains with the L3Out in tenant `common`. If you had a tenant-specific L3Out, it would still be associated to your bridge domains in your respective tenants.
- Two Layer 3 Outs can be in two different VRFs, and they can successfully exchange routes.
- This enhancement is similar to the Application EPG to Layer 3 Out inter-VRF communications. The only difference is that instead of an Application EPG there is another Layer 3 Out. Therefore, in this case, the contract is between two Layer 3 Outs.

In the following figure, there are two Layer 3 Outs with a shared subnet. There is a contract between the Layer 3 external instance profile (l3extInstP) in both the VRFs. In this case, the Shared Layer 3 Out for VRF1 can communicate with the Shared Layer 3 Out for VRF2.

Figure 44: Shared Layer 3 Outs Communicating Between Two VRFs



Configuring Two Shared Layer 3 Outs in Two VRFs Using REST API

The following REST API configuration example that displays how two shared Layer 3 Outs in two VRFs communicate.

Step 1 Configure the provider Layer 3 Out.

Example:

```

<tenant name="t1_provider">
<fvCtx name="VRF1">
<l3extOut name="T0-o1-L3OUT-1">
  <l3extRsEctx tnFvCtxName="o1"/>
  <ospfExtP areaId='60'/>
  <l3extInstP name="l3extInstP-1">
  <fvRsProv tnVzBrCPName="vzBrCP-1">
  </fvRsProv>
  <l3extSubnet ip="192.168.2.0/24" scope="shared-rtctrl, shared-security" aggregate=""/>
  </l3extInstP>
</l3extOut>
</tenant>
  
```

Step 2 Configure the consumer Layer 3 Out.

Example:

```

<tenant name="t1_consumer">
<fvCtx name="VRF2">
<l3extOut name="T0-o1-L3OUT-1">
  <l3extRsEctx tnFvCtxName="o1"/>
  <ospfExtP areaId='70'/>
  <l3extInstP name="l3extInstP-2">
  <fvRsCons tnVzBrCPName="vzBrCP-1">
  </fvRsCons>
  
```

```

        <l3extSubnet ip="199.16.2.0/24" scope="shared-rtctrl, shared-security"
aggregate=""/>
    </l3extInstP>
</l3extOut>
</tenant>

```

Overview Interleak Redistribution for MP-BGP

This topic provides how to configure an interleak redistribution in the Cisco Application Centric Infrastructure (ACI) fabric using Cisco Application Policy Infrastructure Controller (APIC).

In Cisco ACI, a border leaf node on which Layer 3 Outsides (L3Outs) are deployed redistributes L3Out routes to the BGP IPv4/IPv6 address family and then to the MP-BGP VPNv4/VPNv6 address family along with the VRF information so that L3Out routes are distributed from a border leaf node to other leaf nodes through the spine nodes. Interleak redistribution in the Cisco ACI fabric refers to this redistribution of L3Out routes to the BGP IPv4/IPv6 address family. By default, interleak happens for all L3Out routes, such as routes learned through dynamic routing protocols, static routes, and directly-connected subnets of L3Out interfaces, except for routes learned through BGP. Routes learned through BGP are already in the BGP IPv4/IPv6 table and are ready to be exported to MP-BGP VPNv4/VPNv6 without interleak.

Interleak redistribution allows users to apply a route-map to redistribute L3Out routes selectively into BGP to control which routes should be visible to other leaf nodes, or to set some attributes to the routes, such as BGP community, preference, metric, and so on. This redistribution enables selective transit routing to be performed on another border leaf node based on the attributes set by the ingress border leaf node or so that other leaf nodes can prefer routes from one border leaf node to another.

Applying a route map to interleak redistribution from OSPF and EIGRP routes has been available in earlier releases.

Configuring Interleak of External Routes Using the REST API

Before you begin

- The tenant, VRF, and bridge domain are created.
- The external routed domain is created.

Configure an interleak of external routes:

Example:

```

<l3extOut descr="" enforceRtctrl="export" name="out1" ownerKey="" ownerTag="" targetDscp="unspecified">
    <l3extLNodeP configIssues="" descr="" name="Lnodep1" ownerKey="" ownerTag="" tag="yellow-green"
targetDscp="unspecified">
        <l3extRsNodeL3OutAtt rtrId="1.2.3.4" rtrIdLoopBack="yes" tDn="topology/pod-1/node-101"/>
        <l3extLIIfP descr="" name="lifp1" ownerKey="" ownerTag="" tag="yellow-green">
            <ospfIfP authKeyId="1" authType="none" descr="" name="">
                <ospfRsIfPol tnOspfIfPolName=""/>
            </ospfIfP>
        </l3extLIIfP>
    </l3extLNodeP>
</l3extOut>

```



```

    <l3extRsNdIfPol tnNdIfPolName=""/>
    <l3extRsIngressQosDppPol tnQosDppPolName=""/>
    <l3extRsEgressQosDppPol tnQosDppPolName=""/>
    <l3extRsPathL3OutAtt addr="12.12.7.16/24" descr="" encap="unknown" encapScope="local"
ifInstT="l3-port" llAddr="::" mac="00:22:BD:F8:19:FF" mode="regular" mtu="inherit"
tDn="topology/pod-1/paths-101/pathep-[eth1/11]" targetDscp="unspecified"/>
    </l3extLIIfP>
  </l3extLNodeP>
  <l3extRsEctx tnFvCtxName="ctx1"/>
  <l3extRsInterleakPol tnRtctrlProfileName="interleak"/>
  <l3extRsL3DomAtt tDn="uni/l3dom-Domain"/>
  <l3extInstP descr="" matchT="AtleastOne" name="InstP1" prio="unspecified"
targetDscp="unspecified">
    <fvRsCustQosPol tnQosCustomPolName=""/>
    <l3extSubnet aggregate="" descr="" ip="14.15.16.0/24" name=""
scope="export-rtctrl,import-security"/>
  </l3extInstP>
  <ospfExtP areaCost="1" areaCtrl="redistribute,summary" areaId="0.0.0.1" areaType="nssa" descr=""/>
</l3extOut>

```

SVI External Encapsulation Scope

About SVI External Encapsulation Scope

In the context of a Layer 3 Out configuration, a switch virtual interfaces (SVI), is configured to provide connectivity between the ACI leaf switch and a router.

By default, when a single Layer 3 Out is configured with SVI interfaces, the VLAN encapsulation spans multiple nodes within the fabric. This happens because the ACI fabric configures the same bridge domain (VXLAN VNI) across all the nodes in the fabric where the Layer 3 Out SVI is deployed as long as all SVI interfaces use the same external encapsulation (SVI) as shown in the figure.

However, when different Layer 3 Outs are deployed, the ACI fabric uses different bridge domains even if they use the same external encapsulation (SVI) as shown in the figure:

Figure 45: Local Scope Encapsulation and One Layer 3 Out

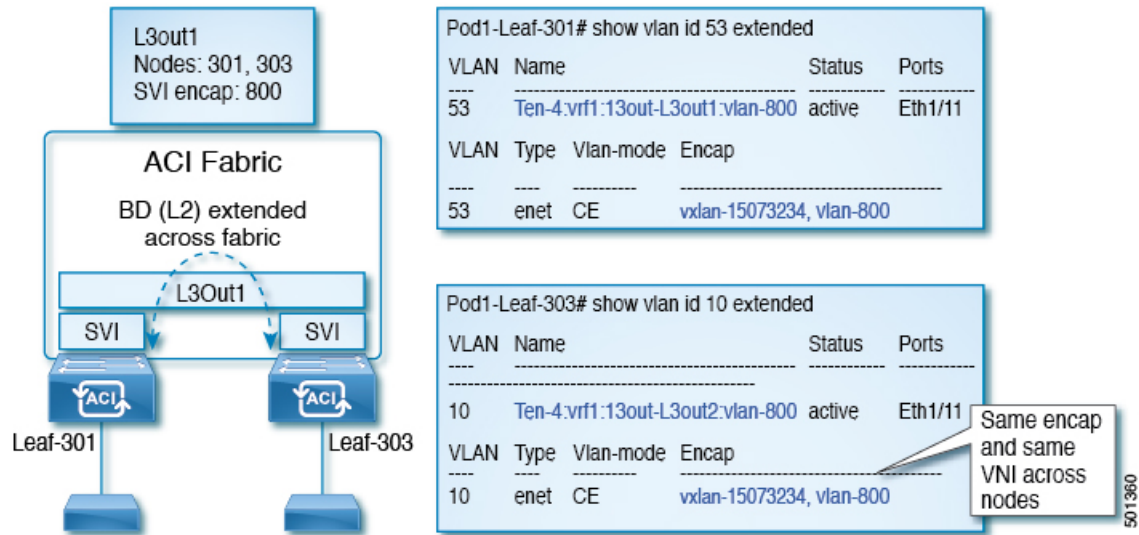
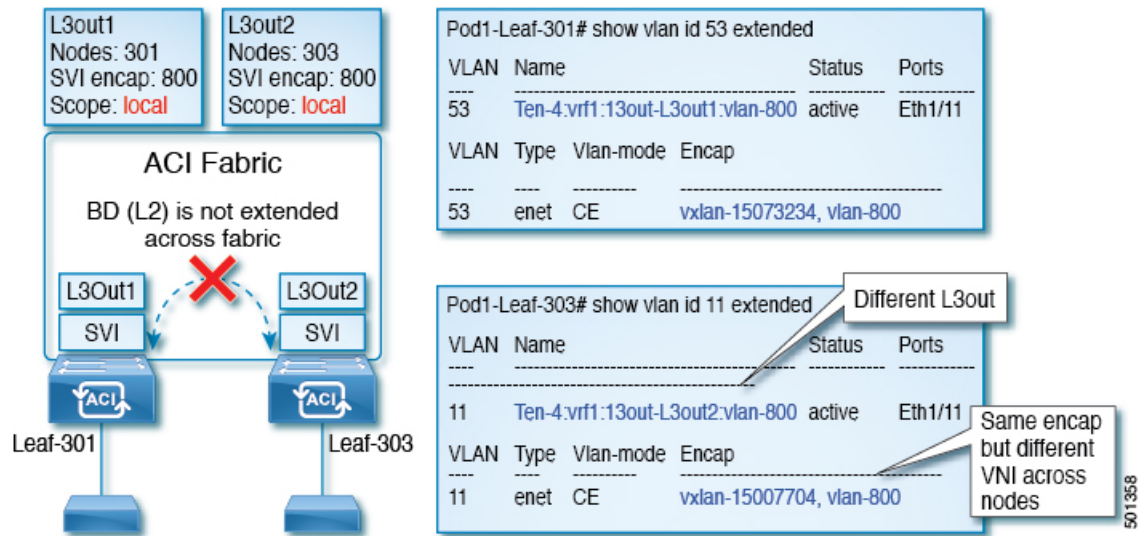


Figure 46: Local Scope Encapsulation and Two Layer 3 Outs

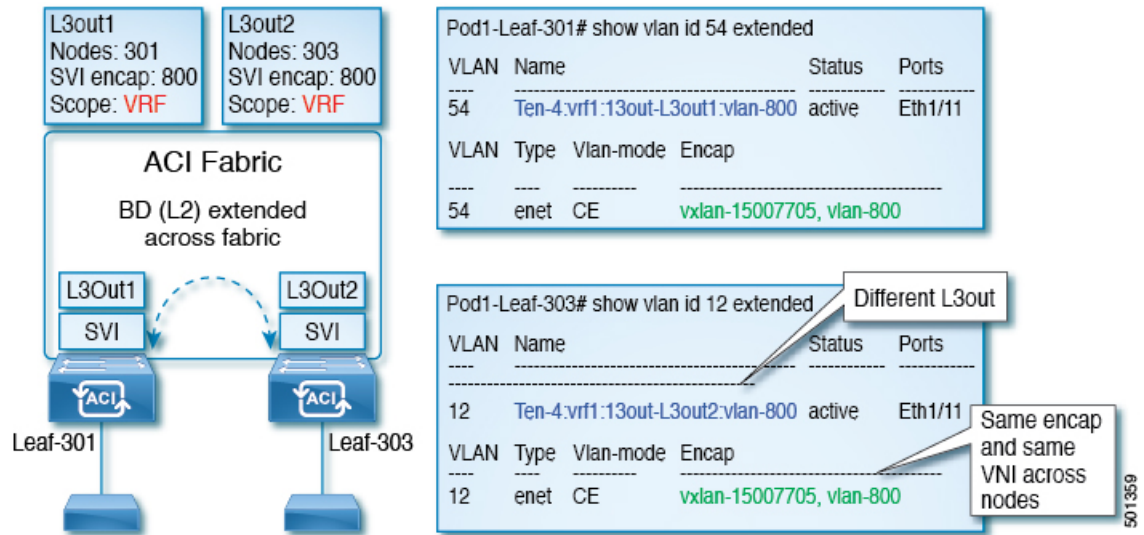


Starting with Cisco APIC release 2.3, it is now possible to choose the behavior when deploying two (or more) Layer 3 Outs using the same external encapsulation (SVI).

The encapsulation scope can now be configured as Local or VRF:

- Local scope (default): The example behavior is displayed in the figure titled *Local Scope Encapsulation and Two Layer 3 Outs*.
- VRF scope: The ACI fabric configures the same bridge domain (VXLAN VNI) across all the nodes and Layer 3 Out where the same external encapsulation (SVI) is deployed. See the example in the figure titled *VRF Scope Encapsulation and Two Layer 3 Outs*.

Figure 47: VRF Scope Encapsulation and Two Layer 3 Outs



Encapsulation Scope Syntax

The options for configuring the scope of the encapsulation used for the Layer 3 Out profile are as follows:

- **Ctx**—The same external SVI in all Layer 3 Outs in the same VRF for a given VLAN encapsulation. This is a global value.
- **Local**—A unique external SVI per Layer 3 Out. This is the default value.

The mapping among the CLI, API, and GUI syntax is as follows:

Table 11: Encapsulation Scope Syntax

CLI	API	GUI
l3out	local	Local
vrf	ctx	VRF



Note The CLI commands to configure encapsulation scope are only supported when the VRF is configured through a named Layer 3 Out configuration.

Configuring SVI Interface Encapsulation Scope Using the REST API

Before you begin

The interface selector is configured.

Configure the SVI interface encapsulation scope.

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- /api/node/mo/.xml -->
<polUni>
  <fvTenant name="coke">
    <l3extOut descr="" dn="uni/tn-coke/out-l3out1" enforceRtctrl="export" name="l3out1" nameAlias=""
ownerKey="" ownerTag="" targetDscp="unspecified">
      <l3extRsL3DomAtt tDn="uni/l3dom-Dom1"/>
      <l3extRsEctx tnFvCtxName="vrf0"/>
      <l3extLNodeP configIssues="" descr="" name="__ui_node_101" nameAlias="" ownerKey="" ownerTag=""
tag="yellow-green" targetDscp="unspecified">
        <l3extRsNodeL3OutAtt rtrId="1.1.1.1" rtrIdLoopBack="no" tDn="topology/pod-1/node-101"/>
        <l3extLIIfP descr="" name="int1_11" nameAlias="" ownerKey="" ownerTag="" tag="yellow-green">
          <l3extRsPathL3OutAtt addr="1.2.3.4/24" descr="" encap="vlan-2001" encapScope="ctx"
ifInstT="ext-svi" llAddr="0.0.0.0" mac="00:22:BD:F8:19:FF" mode="regular" mtu="inherit"
tDn="topology/pod-1/paths-101/pathep-[eth1/5]" targetDscp="unspecified"/>
          <l3extRsNdIfPol tnNdIfPolName=""/>
          <l3extRsIngressQosDppPol tnQosDppPolName=""/>
          <l3extRsEgressQosDppPol tnQosDppPolName=""/>
        </l3extLIIfP>
      </l3extLNodeP>
      <l3extInstP descr="" matchT="AtleastOne" name="epg1" nameAlias="" prefGrMemb="exclude"
prio="unspecified" targetDscp="unspecified">
        <l3extSubnet aggregate="" descr="" ip="101.10.10.1/24" name="" nameAlias=""
scope="import-security"/>
        <fvRsCustQosPol tnQosCustomPolName=""/>
      </l3extInstP>
    </l3extOut>
  </fvTenant>
</polUni>
```

SVI Auto State

About SVI Auto State



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

The Switch Virtual Interface (SVI) represents a logical interface between the bridging function and the routing function of a VLAN in the device. SVI can have members that are physical ports, direct port channels, or virtual port channels. The SVI logical interface is associated with VLANs, and the VLANs have port membership.

The SVI state does not depend on the members. The default auto state behavior for SVI in Cisco APIC is that it remains in the up state when the auto state value is disabled. This means that the SVI remains active even if no interfaces are operational in the corresponding VLAN/s.

If the SVI auto state value is changed to enabled, then it depends on the port members in the associated VLANs. When a VLAN interface has multiple ports in the VLAN, the SVI goes to the down state when all the ports in the VLAN go down.

Table 12: SVI Auto State

SVI Auto State	Description of SVI State
Disabled	SVI remains in the up state even if no interfaces are operational in the corresponding VLAN/s. Disabled is the default SVI auto state value.
Enabled	SVI depends on the port members in the associated VLANs. When a VLAN interface contains multiple ports, the SVI goes into the down state when all the ports in the VLAN go down.

Guidelines and Limitations for SVI Auto State Behavior

Read the following guidelines:

- When you enable or disable the auto state behavior for SVI, you configure the auto state behavior per SVI. There is no global command.

Configuring SVI Auto State Using the REST API

Before you begin

- The tenant and VRF configured.
- A Layer 3 Out is configured and a logical node profile and a logical interface profile under the Layer 3 Out is configured.

Enable the SVI auto state value.

Example:

```
<fvTenant name="t1" >
  <l3extOut name="out1">
    <l3extLNodeP name="__ui_node_101" >
      <l3extLIIfP descr="" name="__ui_eth1_10_vlan_99_af_ipv4" >
        <l3extRsPathL3OutAtt addr="19.1.1.1/24" autostate="enabled" descr="" encap="vlan-100"
encapScope="local" ifInstT="ext-svi" llAddr="::" mac="00:22:BD:F8:19:FF" mode="regular" mtu="inherit"
tDn="topology/pod-1/paths-101/pathep-[eth1/10]" targetDscp="unspecified" />
      </l3extLIIfP>
    </l3extLNodeP>
  </l3extOut>
</fvTenant>
```

To disable the autostate, you must change the value to disabled in the above example. For example, autostate="disabled".

Routing Protocols

BGP and BFD

Guidelines for Configuring a BGP Layer 3 Outside Network Connection

When configuring a BGP external routed network, follow these guidelines:

- The BGP direct route export behavior changed after release 3.2(1), where ACI does not evaluate the originating route type (such as static, direct, and so on) when matching export route map clauses. As a result, the "match direct" deny clause that is always included in the outbound neighbor route map no longer matches direct routes, and direct routes are now advertised based on whether or not a user-defined route map clause matches.

Therefore, the direct route must be advertised explicitly through the route map. Failure to do so will implicitly deny the direct route being advertised.

- The **AS override** option in the **BGP Controls** field in the BGP Peer Connectivity Profile for an L3Out was introduced in release 3.1(2). It allows Cisco Application Centric Infrastructure (ACI) to overwrite a remote AS in the AS_PATH with ACI BGP AS. In Cisco ACI, it is typically used when performing transit routing from an eBGP L3Out to another eBGP L3Out with the same AS number.

However, an issue arises if you enable the **AS override** option when the eBGP neighbor has a different AS number. In this situation, strip the peer-as from the AS_PATH when reflecting it to a peer.

- The **Local-AS Number** option in the BGP Peer Connectivity Profile is supported only for eBGP peering. This enables Cisco ACI border leaf switches to appear to be a member of another AS in addition to its real AS assigned to the fabric MP-BGP Route Reflector Policy. This means that the local AS number must be different from the real AS number of the Cisco ACI fabric. When this feature is configured, Cisco ACI border leaf switches prepend the local AS number to the AS_PATH of the incoming updates and append the same to the AS_PATH of the outgoing updates. Prepending of the local AS number to the incoming updates can be disabled by the **no-prepend** setting in the **Local-AS Number Config**. The **no-prepend + replace-as** setting can be used to prevent the local AS number from being appended to the outgoing updates in addition to not prepending the same to the incoming updates.
- A router ID for an L3Out for any routing protocols cannot be the same IP address or the same subnet as the L3Out interfaces such as routed interface, sub-interface or SVI. However, if needed, a router ID can be the same as one of the L3Out loopback IP addresses.
- If you have multiple L3Outs of the same routing protocol on the same leaf switch in the same VRF instance, the router ID for those must be the same. If you need a loopback with the same IP address as the router ID, you can configure the loopback in only one of those L3Outs.
- There are two ways to define the BGP peer for an L3Out:
 - Through the BGP peer connectivity profile (**bgpPeerP**) at the logical node profile level (**l3extLNodeP**), which associates the BGP peer to the loopback IP address. When the BGP peer is configured at this level, a loopback address is expected for BGP connectivity, so a fault is raised if the loopback address configuration is missing.
 - Through the BGP peer connectivity profile (**bgpPeerP**) at the logical interface profile level (**l3extRsPathL3OutAtt**), which associates the BGP peer to the respective interface or sub-interface.

- You must configure an IPv6 address to enable peering over loopback using IPv6.
- Tenant networking protocol policies for BGP `l3extOut` connections can be configured with a maximum prefix limit that enables monitoring and restricting the number of route prefixes received from a peer. After the maximum prefix limit is exceeded, a log entry can be recorded, further prefixes can be rejected, the connection can be restarted if the count drops below the threshold in a fixed interval, or the connection is shut down. You can use only one option at a time. The default setting is a limit of 20,000 prefixes, after which new prefixes are rejected. When the reject option is deployed, BGP accepts one more prefix beyond the configured limit and the Cisco Application Policy Infrastructure Controller (APIC) raises a fault.



Note Cisco ACI does not support IP fragmentation. Therefore, when you configure Layer 3 Outside (L3Out) connections to external routers, or Multi-Pod connections through an Inter-Pod Network (IPN), it is recommended that the interface MTU is set appropriately on both ends of a link. On some platforms, such as Cisco ACI, Cisco NX-OS, and Cisco IOS, the configurable MTU value does not take into account the Ethernet headers (matching IP MTU, and excluding the 14-18 Ethernet header size), while other platforms, such as IOS-XR, include the Ethernet header in the configured MTU value. A configured value of 9000 results in a max IP packet size of 9000 bytes in Cisco ACI, Cisco NX-OS, and Cisco IOS, but results in a max IP packet size of 8986 bytes for an IOS-XR untagged interface.

For the appropriate MTU values for each platform, see the relevant configuration guides.

We highly recommend that you test the MTU using CLI-based commands. For example, on the Cisco NX-OS CLI, use a command such as `ping 1.1.1.1 df-bit packet-size 9000 source-interface ethernet 1/1`.

BGP Connection Types and Loopback Guidelines

The ACI supports the following BGP connection types and summarizes the loopback guidelines for them:

BGP Connection Type	Loopback required	Loopback same as Router ID	Static/OSPF route required
iBGP direct	No	Not applicable	No
iBGP loopback peering	Yes, a separate loopback per L3Out	No, if multiple Layer 3 out are on the same node	Yes
eBGP direct	No	Not applicable	No
eBGP loopback peering (multi-hop)	Yes, a separate loopback per L3Out	No, if multiple Layer 3 out are on the same node	Yes

Per VRF Per Node BGP Timer Values

Prior to the introduction of this feature, for a given VRF, all nodes used the same BGP timer values.

With the introduction of the per VRF per node BGP timer values feature, BGP timers can be defined and associated on a per VRF per node basis. A node can have multiple VRFs, each corresponding to a `fvCtx`. A node configuration (`l3extLNodeP`) can now contain configuration for BGP Protocol Profile (`bgpProtP`) which in turn refers to the desired BGP Context Policy (`bgpCtxPol`). This makes it possible to have a different node within the same VRF contain different BGP timer values.

For each VRF, a node has a `bgpDom` concrete MO. Its name (primary key) is the VRF, `<fvTenant>:<fvCtx>`. It contains the BGP timer values as attributes (for example, `holdIntvl`, `kaIntvl`, `maxAsLimit`).

All the steps necessary to create a valid Layer 3 Out configuration are required to successfully apply a per VRF per node BGP timer. For example, MOs such as the following are required: `fvTenant`, `fvCtx`, `l3extOut`, `l3extInstP`, `LNodeP`, `bgpRR`.

On a node, the BGP timer policy is chosen based on the following algorithm:

- If `bgpProtP` is specified, then use `bgpCtxPol` referred to under `bgpProtP`.
- Else, if specified, use `bgpCtxPol` referred to under corresponding `fvCtx`.
- Else, if specified, use the default policy under the tenant, for example, `uni/tn-<tenant>/bgpCtxP-default`.
- Else, use the default policy under tenant `common`, for example, `uni/tn-common/bgpCtxP-default`. This one is pre-programmed.

Configuring an MP-BGP Route Reflector Using the REST API

Step 1 Mark the spine switches as route reflectors.

Example:

```
POST https://apic-ip-address/api/policymgr/mo/uni/fabric.xml
```

```
<bgpInstPol name="default">
  <bgpAsP asn="1" />
  <bgpRRP>
    <bgpRRNodePEp id="<spine_id1>" />
    <bgpRRNodePEp id="<spine_id2>" />
  </bgpRRP>
</bgpInstPol>
```

Step 2 Set up the pod selector using the following post.

Example:

For the FuncP setup—

```
POST https://apic-ip-address/api/policymgr/mo/uni.xml
```

```
<fabricFuncP>
  <fabricPodPGrp name="bgpRRPodGrp">
    <fabricRsPodPGrpBGPRRP tnBgpInstPolName="default" />
  </fabricPodPGrp>
</fabricFuncP>
```

Example:

For the PodP setup—


```
POST https://apic-ip-address/api/policymgr/mo/uni.xml

<fabricPodP name="default">
  <fabricPodS name="default" type="ALL">
    <fabricRsPodPGrp tDn="uni/fabric/funcprof/podpgrp-bgpRRPodGrp"/>
  </fabricPodS>
</fabricPodP>
```

Configuring BGP External Routed Network Using the REST API

Before you begin

The tenant where you configure the BGP external routed network is already created.

The following shows how to configure the BGP external routed network using the REST API:

For Example:

Example:

```
<l3extOut descr="" dn="uni/tn-t1/out-l3out-bgp" enforceRtctrl="export" name="l3out-bgp" ownerKey=""
ownerTag="" targetDscp="unspecified">
  <l3extRsEctx tnFvCtxName="ctx3"/>
  <l3extLNodeP configIssues="" descr="" name="l3extLNodeP_1" ownerKey="" ownerTag="" tag="yellow-green"
targetDscp="unspecified">
  <l3extRsNodeL3OutAtt rtrId="1.1.1.1" rtrIdLoopBack="no" tDn="topology/pod-1/node-101"/>
  <l3extLIfP descr="" name="l3extLIfP_2" ownerKey="" ownerTag="" tag="yellow-green">
  <l3extRsNdIfPol tnNdIfPolName=""/>
  <l3extRsIngressQosDppPol tnQosDppPolName=""/>
  <l3extRsEgressQosDppPol tnQosDppPolName=""/>
  <l3extRsPathL3OutAtt addr="3001::31:0:1:2/120" descr="" encap="vlan-3001" encapScope="local"
ifInstT="sub-interface" llAddr="::" mac="00:22:BD:F8:19:FF" mode="regular" mtu="inherit"
tDn="topology/pod-1/paths-101/pathep-[eth1/8]" targetDscp="unspecified">
  <bgpPeerP addr="3001::31:0:1:0/120" allowedSelfAsCnt="3" ctrl="send-com,send-ext-com" descr=""
name="" peerCtrl="bfd" privateASctrl="remove-all,remove-exclusive,replace-as" ttl="1" weight="100">

    <bgpRsPeerPfxPol tnBgpPeerPfxPolName=""/>
    <bgpAsP asn="3001" descr="" name=""/>
  </bgpPeerP>
</l3extRsPathL3OutAtt>
</l3extLIfP>
<l3extLIfP descr="" name="l3extLIfP_1" ownerKey="" ownerTag="" tag="yellow-green">
  <l3extRsNdIfPol tnNdIfPolName=""/>
  <l3extRsIngressQosDppPol tnQosDppPolName=""/>
  <l3extRsEgressQosDppPol tnQosDppPolName=""/>
  <l3extRsPathL3OutAtt addr="31.0.1.2/24" descr="" encap="vlan-3001" encapScope="local"
ifInstT="sub-interface" llAddr="::" mac="00:22:BD:F8:19:FF" mode="regular" mtu="inherit"
tDn="topology/pod-1/paths-101/pathep-[eth1/8]" targetDscp="unspecified">
  <bgpPeerP addr="31.0.1.0/24" allowedSelfAsCnt="3" ctrl="send-com,send-ext-com" descr="" name=""
peerCtrl="" privateASctrl="remove-all,remove-exclusive,replace-as" ttl="1" weight="100">
    <bgpRsPeerPfxPol tnBgpPeerPfxPolName=""/>
    <bgpLocalAsnP asnPropagate="none" descr="" localAsn="200" name=""/>
    <bgpAsP asn="3001" descr="" name=""/>
  </bgpPeerP>
</l3extRsPathL3OutAtt>
</l3extLIfP>
</l3extLNodeP>
<l3extRsL3DomAtt tDn="uni/l3dom-l3-dom"/>
```

```

<l3extRsDampeningPol af="ipv6-ucast" tnRtctrlProfileName="damp_rp"/>
<l3extRsDampeningPol af="ipv4-ucast" tnRtctrlProfileName="damp_rp"/>
<l3extInstP descr="" matchT="AtleastOne" name="l3extInstP_1" prio="unspecified"
targetDscp="unspecified">
  <l3extSubnet aggregate="" descr="" ip="130.130.130.0/24" name="" scope="import-rtctrl"></l3extSubnet>

  <l3extSubnet aggregate="" descr="" ip="130.130.131.0/24" name="" scope="import-rtctrl"/>
  <l3extSubnet aggregate="" descr="" ip="120.120.120.120/32" name=""
scope="export-rtctrl,import-security"/>
  <l3extSubnet aggregate="" descr="" ip="3001::130:130:130:100/120" name="" scope="import-rtctrl"/>
</l3extInstP>
<bgpExtP descr=""/>
</l3extOut>
<rtctrlProfile descr="" dn="uni/tn-t1/prof-damp_rp" name="damp_rp" ownerKey="" ownerTag=""
type="combinable">
  <rtctrlCtxP descr="" name="ipv4_rpc" order="0">
    <rtctrlScope descr="" name="">
      <rtctrlRsScopeToAttrP tnRtctrlAttrPName="act_rule"/>
    </rtctrlScope>
  </rtctrlCtxP>
</rtctrlProfile>
<rtctrlAttrP descr="" dn="uni/tn-t1/attr-act_rule" name="act_rule">
  <rtctrlSetDamp descr="" halfLife="15" maxSuppressTime="60" name="" reuse="750" suppress="2000"
type="dampening-pol"/>
</rtctrlAttrP>

```

Configuring BFD Consumer Protocols Using the REST API

Step 1 The following example shows the interface configuration for bidirectional forwarding detection (BFD):

Example:

```

<fvTenant name="ExampleCorp">
  <bfdIfPol name="bfdIfPol" minTxIntvl="400" minRxIntvl="400" detectMult="5" echoRxIntvl="400"
echoAdminSt="disabled"/>
  <l3extOut name="l3-out">
    <l3extLNodeP name="leaf1">
      <l3extRsNodeL3OutAtt tDn="topology/pod-1/node-101" rtrId="2.2.2.2"/>
      <l3extLIIfP name='portIpv4'>
        <l3extRsPathL3OutAtt tDn="topology/pod-1/paths-101/pathep-[eth1/11]" ifInstT='l3-port'
addr="10.0.0.1/24" mtu="1500"/>
        <bfdIfP type="sha1" key="password">
          <bfdRsIfPol tnBfdIfPolName='bfdIfPol' />
        </bfdIfP>
      </l3extLIIfP>
    </l3extLNodeP>
  </l3extOut>
</fvTenant>

```

Step 2 The following example shows the interface configuration for enabling BFD on OSPF and EIGRP:

Example:

BFD on leaf switch

```

<fvTenant name="ExampleCorp">
  <ospfIfPol name="ospf_intf_pol" cost="10" ctrl="bfd"/>

```

```
<eigrpIfPol ctrl="nh-self,split-horizon,bfd" dn="uni/tn-Coke/eigrpIfPol-eigrp_if_default"
</fvTenant>
```

Example:

BFD on spine switch

```
<l3extLNodeP name="bSpine">
  <l3extRsNodeL3OutAtt tDn="topology/pod-1/node-103" rtrId="192.3.1.8">
    <l3extLoopBackIfP addr="10.10.3.1" />
    <l3extInfraNodeP fabricExtCtrlPeering="false" />
  </l3extRsNodeL3OutAtt>
  <l3extLIIfP name='portIf'>
    <l3extRsPathL3OutAtt tDn="topology/pod-1/paths-103/pathep-[eth5/10]" encap='vlan-4'
ifInstT='sub-interface' addr="20.3.10.1/24"/>
    <ospfIfP>
      <ospfRsIfPol tnOspfIfPolName='ospf_intf_pol' />
    </ospfIfP>
    <bfdIfP name="test" type="sha1" key="hello" status="created,modified">
      <bfdRsIfPol tnBfdIfPolName='default' status="created,modified"/>
    </bfdIfP>
  </l3extLIIfP>
</l3extLNodeP>
```

Step 3 The following example shows the interface configuration for enabling BFD on BGP:

Example:

```
<fvTenant name="ExampleCorp">
  <l3extOut name="l3-out">
    <l3extLNodeP name="leaf1">
      <l3extRsNodeL3OutAtt tDn="topology/pod-1/node-101" rtrId="2.2.2.2"/>
      <l3extLIIfP name='portIpv4'>
        <l3extRsPathL3OutAtt tDn="topology/pod-1/paths-101/pathep-[eth1/11]" ifInstT='l3-port'
addr="10.0.0.1/24" mtu="1500">
          <bgpPeerP addr="4.4.4.4/24" allowedSelfAsCnt="3" ctrl="bfd" descr="" name=""
peerCtrl="" ttl="1">
            <bgpRsPeerPfxPol tnBgpPeerPfxPolName="" />
            <bgpAsP asn="3" descr="" name="" />
          </bgpPeerP>
        </l3extRsPathL3OutAtt>
      </l3extLIIfP>
    </l3extLNodeP>
  </l3extOut>
</fvTenant>
```

Step 4 The following example shows the interface configuration for enabling BFD on Static Routes:

Example:

BFD on leaf switch

```
<fvTenant name="ExampleCorp">
  <l3extOut name="l3-out">
    <l3extLNodeP name="leaf1">
      <l3extRsNodeL3OutAtt tDn="topology/pod-1/node-101" rtrId="2.2.2.2">
        <ipRouteP ip="192.168.3.4" rtCtrl="bfd">
          <ipNextHopP nhAddr="192.168.62.2"/>
        </ipRouteP>
      </l3extRsNodeL3OutAtt>
    </l3extLNodeP>
  </l3extOut>
</fvTenant>
```

```

    </l3extRsNodeL3OutAtt>
    <l3extLIIfP name='portIpv4'>
      <l3extRsPathL3OutAtt tDn="topology/pod-1/paths-101/pathep-[eth1/3]" ifInstT='l3-port'
        addr="10.10.10.2/24" mtu="1500" status="created,modified" />
    </l3extLIIfP>

  </l3extLNodeP>

</l3extOut>
</fvTenant>

```

Example:**BFD on spine switch**

```

<l3extLNodeP name="bSpine">

  <l3extRsNodeL3OutAtt tDn="topology/pod-1/node-103" rtrId="192.3.1.8">
    <ipRouteP ip="0.0.0.0" rtCtrl="bfd">
      <ipNextHopP nhAddr="192.168.62.2"/>
    </ipRouteP>
  </l3extRsNodeL3OutAtt>

  <l3extLIIfP name='portIf'>
    <l3extRsPathL3OutAtt tDn="topology/pod-1/paths-103/pathep-[eth5/10]" encap='vlan-4'
      ifInstT='sub-interface' addr="20.3.10.1/24"/>

    <bfdIfP name="test" type="shal" key="hello" status="created,modified">
      <bfdRsIfPol tnBfdIfPolName='default' status="created,modified"/>
    </bfdIfP>
  </l3extLIIfP>

</l3extLNodeP>

```

Step 5 The following example shows the interface configuration for enabling BFD on IS-IS:

Example:

```

<fabricInst>
  <l3IfPol name="testL3IfPol" bfdIsis="enabled"/>
  <fabricLeafP name="LeNode" >
    <fabricRsLePortP tDn="uni/fabric/leportp-leaf_profile" />
    <fabricLeafS name="spsw" type="range">
    <fabricNodeBlk name="node101" to_"=102" from_"=101" />
  </fabricLeafS>
  </fabricLeafP>

  <fabricSpineP name="SpNode" >
    <fabricRsSpPortP tDn="uni/fabric/spportp-spine_profile" />
    <fabricSpineS name="spsw" type="range">
      <fabricNodeBlk name="node103" to_"=103" from_"=103" />
    </fabricSpineS>
  </fabricSpineP>

  <fabricLePortP name="leaf_profile">
    <fabricLFPortS name="leafIf" type="range">
    <fabricPortBlk name="spBlk" fromCard="1" fromPort="49" toCard="1" toPort="49" />
      <fabricRsLePortPGrp tDn="uni/fabric/funcprof/leportgrp-LeTestPGrp" />
    </fabricLFPortS>
  </fabricLePortP>

  <fabricSpPortP name="spine_profile">
    <fabricSFPortS name="spineIf" type="range">
      <fabricPortBlk name="spBlk" fromCard="5" fromPort="1" toCard="5" toPort="2" />
    </fabricSFPortS>
  </fabricSpPortP>

```

```

    <fabricRsSpPortPGrp tDn="uni/fabric/funcprof/spportgrp-SpTestPGrp" />
  </fabricSFPortS>
  </fabricSpPortP>

  <fabricFuncP>
    <fabricLePortPGrp name = "LeTestPGrp">
  <fabricRsL3IfPol tnL3IfPolName="testL3IfPol"/>
    </fabricLePortPGrp>

    <fabricSpPortPGrp name = "SpTestPGrp">
  <fabricRsL3IfPol tnL3IfPolName="testL3IfPol"/>
    </fabricSpPortPGrp>

</fabricFuncP>

</fabricInst>

```

Configuring BFD Globally Using the REST API

The following REST API shows the global configuration for bidirectional forwarding detection (BFD):

Example:

```

<polUni>
  <infraInfra>
    <bfdIpv4InstPol name="default" echoSrcAddr="1.2.3.4" slowIntvl="1000" minTxIntvl="150"
minRxIntvl="250" detectMult="5" echoRxIntvl="200"/>
    <bfdIpv6InstPol name="default" echoSrcAddr="34::1/64" slowIntvl="1000" minTxIntvl="150"
minRxIntvl="250" detectMult="5" echoRxIntvl="200"/>
  </infraInfra>
</polUni>

```

Configuring BFD Interface Override Using the REST API

The following REST API shows the interface override configuration for bidirectional forwarding detection (BFD):

Example:

```

<fvTenant name="ExampleCorp">
  <bfdIfPol name="bfdIfPol" minTxIntvl="400" minRxIntvl="400" detectMult="5" echoRxIntvl="400"
echoAdminSt="disabled"/>
  <l3extOut name="l3-out">
    <l3extLNodeP name="leaf1">
      <l3extRsNodeL3OutAtt tDn="topology/pod-1/node-101" rtrId="2.2.2.2"/>
      <l3extLIfP name='portIpv4'>
        <l3extRsPathL3OutAtt tDn="topology/pod-1/paths-101/pathep-[eth1/11]" ifInstT='l3-port'
addr="10.0.0.1/24" mtu="1500"/>
        <bfdIfP type="sha1" key="password">
          <bfdRsIfPol tnBfdIfPolName='bfdIfPol' />
        </bfdIfP>
      </l3extLIfP>
    </l3extLNodeP>
  </l3extOut>
</fvTenant>

```

```
</l3extOut>
</fvTenant>
```

Configuring a Per VRF Per Node BGP Timer Using the REST API

The following example shows how to configure Per VRF Per node BGP timer in a node. Configure `bgpProtP` under `l3extLNodeP` configuration. Under `bgpProtP`, configure a relation (`bgpRsBgpNodeCtxPol`) to the desired BGP Context Policy (`bgpCtxPol`).

Configure a node specific BGP timer policy on `node1`, and configure `node2` with a BGP timer policy that is not node specific.

Example:

```
POST https://apic-ip-address/mo.xml
```

```
<fvTenant name="tn1" >
  <bgpCtxPol name="pol1" staleIntvl="25" />
  <bgpCtxPol name="pol2" staleIntvl="35" />
  <fvCtx name="ctx1" >
    <fvRsBgpCtxPol tnBgpCtxPolName="pol1"/>
  </fvCtx>
  <l3extout name="out1" >
    <l3extRsEctx toFvCtxName="ctx1" />
    <l3extLNodeP name="node1" >
      <bgpProtP name="protpl" >
        <bgpRsBgpNodeCtxPol tnBgpCtxPolName="pol2" />
      </bgpProtP>
    </l3extLNodeP>
    <l3extLNodeP name="node2" >
    </l3extLNodeP>
```

In this example, `node1` gets BGP timer values from policy `pol2`, and `node2` gets BGP timer values from `pol1`. The timer values are applied to the `bgpDom` corresponding to VRF `tn1:ctx1`. This is based upon the BGP timer policy that is chosen following the algorithm described in the *Per VRF Per Node BGP Timer Values* section.

Deleting a Per VRF Per Node BGP Timer Using the REST API

The following example shows how to delete an existing Per VRF Per node BGP timer in a node.

Delete the node specific BGP timer policy on `node1`.

Example:

```
POST https://apic-ip-address/mo.xml
```

```
<fvTenant name="tn1" >
  <bgpCtxPol name="pol1" staleIntvl="25" />
  <bgpCtxPol name="pol2" staleIntvl="35" />
  <fvCtx name="ctx1" >
    <fvRsBgpCtxPol tnBgpCtxPolName="pol1"/>
  </fvCtx>
  <l3extout name="out1" >
    <l3extRsEctx toFvCtxName="ctx1" />
    <l3extLNodeP name="node1" >
      <bgpProtP name="protpl" status="deleted" >
```

```

    <bgpRsBgpNodeCtxPol tnBgpCtxPolName="pol2" />
  </bgpProtP>
</l3extLNodeP>
<l3extLNodeP name="node2" >
</l3extLNodeP>

```

The code phrase `<bgpProtP name="protpl" status="deleted" >` in the example above, deletes the BGP timer policy. After the deletion, `node1` defaults to the BGP timer policy for the VRF with which `node1` is associated, which is `pol1` in the above example.

Configuring BGP Max Path

The following feature enables you to add the maximum number of paths to the route table to enable equal cost, multipath load balancing.

Configuring BGP Max Path Using the REST API

This following example provides information on how to configure the BGP Max Path feature using the REST API:

```

<fvTenant descr="" dn="uni/tn-t1" name="t1">
  <fvCtx name="v1">
    <fvRsCtxToBgpCtxAfPol af="ipv4-ucast" tnBgpCtxAfPolName="bgpCtxPol1"/>
  </fvCtx>
  <bgpCtxAfPol name="bgpCtxPol1" maxEcmp="8" maxEcmpIbgp="4"/>
</fvTenant>

```

Configuring AS Path Prepend

A BGP peer can influence the best-path selection by a remote peer by increasing the length of the AS-Path attribute. AS-Path Prepend provides a mechanism that can be used to increase the length of the AS-Path attribute by prepending a specified number of AS numbers to it.

AS-Path prepending can only be applied in the outbound direction using route-maps. AS Path prepending does not work in iBGP sessions.

The AS Path Prepend feature enables modification as follows:

Prepend	<p>Appends the specified AS number to the AS path of the route matched by the route map.</p> <p>Note</p> <ul style="list-style-type: none"> You can configure more than one AS number. 4 byte AS numbers are supported. You can prepend a total 32 AS numbers. You must specify the order in which the AS Number is inserted into the AS Path attribute.
Prepend-last-as	Prepends the last AS numbers to the AS path with a range between 1 and 10.

The following table describes the selection criteria for implementation of AS Path Prepend:

Prepend	1	Prepend the specified AS number.
Prepend-last-as	2	Prepend the last AS numbers to the AS path.
DEFAULT	Prepend(1)	Prepend the specified AS number.

Configuring AS Path Prepend Using the REST API

The following example provides information on how to configure the AS Path Prepend feature using the REST API:

```
<?xml version="1.0" encoding="UTF-8"?>
<fvTenant name="coke">
  <rtctrlAttrP name="attrp1">
    <rtctrlSetASPath criteria="prepend">
      <rtctrlSetASPathASN asn="100" order="1"/>
      <rtctrlSetASPathASN asn="200" order="10"/>
      <rtctrlSetASPathASN asn="300" order="5"/>
    </rtctrlSetASPath/>
    <rtctrlSetASPath criteria="prepend-last-as" lastnum="9" />
  </rtctrlAttrP>

  <l3extOut name="out1">
    <rtctrlProfile name="rp1">
      <rtctrlCtxP name="ctxp1" order="1">
        <rtctrlScope>
          <rtctrlRsScopeToAttrP tnRtctrlAttrPName="attrp1"/>
        </rtctrlScope>
      </rtctrlCtxP>
    </rtctrlProfile>
  </l3extOut>
</fvTenant>
```

About BGP Autonomous System Override

Loop prevention in BGP is done by verifying the Autonomous System number in the Autonomous System Path. If the receiving router sees its own Autonomous System number in the Autonomous System path of the received BGP packet, the packet is dropped. The receiving router assumes that the packet originated from its own Autonomous System and has reached the same place from where it originated initially. This setting is the default to prevent route loops from occurring.

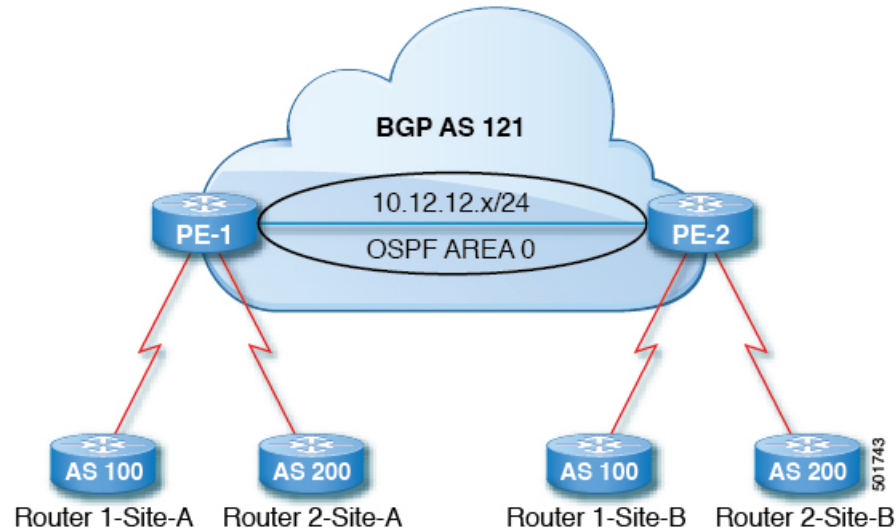
The default setting to prevent route loops from occurring could create an issue if you use the same Autonomous System number along various sites and disallow user sites with identical Autonomous System numbers to link by another Autonomous System number. In such a scenario, routing updates from one site is dropped when the other site receives them.

To prevent such a situation from occurring, beginning with the Cisco APIC Release 3.1(2m), you can now enable the BGP Autonomous System override feature to override the default setting. You must also enable the Disable Peer AS Check at the same time.

The Autonomous System override function replaces the Autonomous System number from the originating router with the Autonomous System number of the sending BGP router in the AS Path of the outbound routes. This feature can be enabled per feature per address family (IPv4 or IPv6).

The Autonomous System Override feature is supported with GOLF Layer 3 configurations and Non-GOLF Layer 3 configurations.

Figure 48: Example Topology Illustrating the Autonomous System Override Process



Router 1 and Router 2 are the two customers with multiple sites (Site-A and Site-B). Customer Router 1 operates under AS 100 and customer Router 2 operates under AS 200.

The above diagram illustrates the Autonomous System (AS) override process as follows:

1. Router 1-Site-A advertises route 10.3.3.3 with AS100.
2. Router PE-1 propagates this as an internal route to PE2 as AS100.
3. Router PE-2 prepends 10.3.3.3 with AS121 (replaces 100 in the AS path with 121), and propagates the prefix.
4. Router 2-Site-B accepts the 10.3.3.3 update.

Configuring BGP External Routed Network with Autonomous System Override Enabled Using the REST API

SUMMARY STEPS

1. Configure the BGP External Routed Network with Autonomous override enabled.

DETAILED STEPS

Configure the BGP External Routed Network with Autonomous override enabled.

Note The line of code that is in bold displays the BGP AS override portion of the configuration. This feature was introduced in the Cisco APIC Release 3.1(2m).

Example:

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

```

<fvCtx name="coke" status="">
  <bgpRtTargetP af="ipv4-ucast">
    <bgpRtTarget type="import" rt="route-target:as4-nn2:1234:1300" />
    <bgpRtTarget type="export" rt="route-target:as4-nn2:1234:1300" />
  </bgpRtTargetP>
  <bgpRtTargetP af="ipv6-ucast">
    <bgpRtTarget type="import" rt="route-target:as4-nn2:1234:1300" />
    <bgpRtTarget type="export" rt="route-target:as4-nn2:1234:1300" />
  </bgpRtTargetP>
</fvCtx>

<fvBD name="cokeBD">
  <!-- Association from Bridge Doamin to Private Network -->
  <fvRsCtx tnFvCtxName="coke" />
  <fvRsBDToOut tnL3extOutName="routAccounting" />
  <!-- Subnet behind the bridge domain-->
  <fvSubnet ip="20.1.1.1/16" scope="public"/>
  <fvSubnet ip="2000:1::1/64" scope="public"/>
</fvBD>

<fvBD name="cokeBD2">
  <!-- Association from Bridge Doamin to Private Network -->
  <fvRsCtx tnFvCtxName="coke" />
  <fvRsBDToOut tnL3extOutName="routAccounting" />
  <!-- Subnet behind the bridge domain-->
  <fvSubnet ip="30.1.1.1/16" scope="public"/>
</fvBD>

<fvBrCP name="webCtrct" scope="global">
  <vzSubj name="http">
    <vzRsSubjFiltAtt tnVzFilterName="default"/>
  </vzSubj>
</fvBrCP>

<!-- GOLF L3Out -->
<l3extOut name="routAccounting">
  <l3extConsLbl name="golf_transit" owner="infra" status="" />
  <bgpExtP/>
  <l3extInstP name="accountingInst">
    <!--
    <l3extSubnet ip="192.2.2.0/24" scope="import-security,import-rtctrl" />
    <l3extSubnet ip="192.3.2.0/24" scope="export-rtctrl"/>
    <l3extSubnet ip="192.5.2.0/24" scope="export-rtctrl"/>
    <l3extSubnet ip="64:ff9b::c007:200/120" scope="export-rtctrl" />
    -->
    <l3extSubnet ip="0.0.0.0/0"
      scope="export-rtctrl,import-security"
      aggregate="export-rtctrl"
    />
    <fvRsProv tnVzBrCPName="webCtrct"/>
  </l3extInstP>

  <l3extRsEctx tnFvCtxName="coke"/>
</l3extOut>

<fvAp name="cokeAp">
  <fvAEPg name="cokeEPg" >
    <fvRsBd tnFvBDName="cokeBD" />
    <fvRsPathAtt tDn="topology/pod-1/paths-103/pathep-[eth1/20]" encap="vlan-100"
instrImedcy="immediate" mode="regular"/>
    <fvRsCons tnVzBrCPName="webCtrct"/>
  </fvAEPg>
  <fvAEPg name="cokeEPg2" >
    <fvRsBd tnFvBDName="cokeBD2" />

```

```

        <fvRsPathAtt tDn="topology/pod-1/paths-103/pathep-[eth1/20]" encap="vlan-110"
instrImedcy="immediate" mode="regular"/>
        <fvRsCons tnVzBrCPName="webCtrct"/>
    </fvAEPg>
</fvAp>

<!-- Non GOLF L3Out-->
<l3extOut name="NonGolfOut">
    <bgpExtP/>
    <l3extLNodeP name="bLeaf">
        <!--
        <l3extRsNodeL3OutAtt tDn="topology/pod-1/node-101" rtrId="20.1.13.1"/>
        -->
        <l3extRsNodeL3OutAtt tDn="topology/pod-1/node-101" rtrId="20.1.13.1">
        <l3extLoopBackIfP addr="1.1.1.1"/>

        <ipRouteP ip="2.2.2.2/32" >
        <ipNextHopP nhAddr="20.1.12.3"/>
    </ipRouteP>

    </l3extRsNodeL3OutAtt>
    <l3extLIfP name='portIfV4'>
        <l3extRsPathL3OutAtt tDn="topology/pod-1/paths-101/pathep-[eth1/17]" encap='vlan-1010'
ifInstT='sub-interface' addr="20.1.12.2/24">

        </l3extRsPathL3OutAtt>
    </l3extLIfP>
    <l3extLIfP name='portIfV6'>
        <l3extRsPathL3OutAtt tDn="topology/pod-1/paths-101/pathep-[eth1/17]" encap='vlan-1010'
ifInstT='sub-interface' addr="64:ff9b::1401:302/120">
        <bgpPeerP addr="64:ff9b::1401:d03" ctrl="send-com,send-ext-com" />
    </l3extRsPathL3OutAtt>
    </l3extLIfP>
    <bgpPeerP addr="2.2.2.2" ctrl="as-override,disable-peer-as-check, send-com,send-ext-com"
status="" />
    </l3extLNodeP>
    <!--
    <bgpPeerP addr="2.2.2.2" ctrl="send-com,send-ext-com" status="" />
    -->
    <l3extInstP name="accountingInst">
        <l3extSubnet ip="192.10.0.0/16" scope="import-security,import-rtctrl" />
        <l3extSubnet ip="192.3.3.0/24" scope="import-security,import-rtctrl" />
        <l3extSubnet ip="192.4.2.0/24" scope="import-security,import-rtctrl" />
        <l3extSubnet ip="64:ff9b::c007:200/120" scope="import-security,import-rtctrl" />
        <l3extSubnet ip="192.2.2.0/24" scope="export-rtctrl" />
        <l3extSubnet ip="0.0.0.0/0"
            scope="export-rtctrl,import-rtctrl,import-security"
            aggregate="export-rtctrl,import-rtctrl"

        />
    </l3extInstP>
    <l3extRsEctx tnFvCtxName="coke"/>
</l3extOut>

</fvTenant>

```

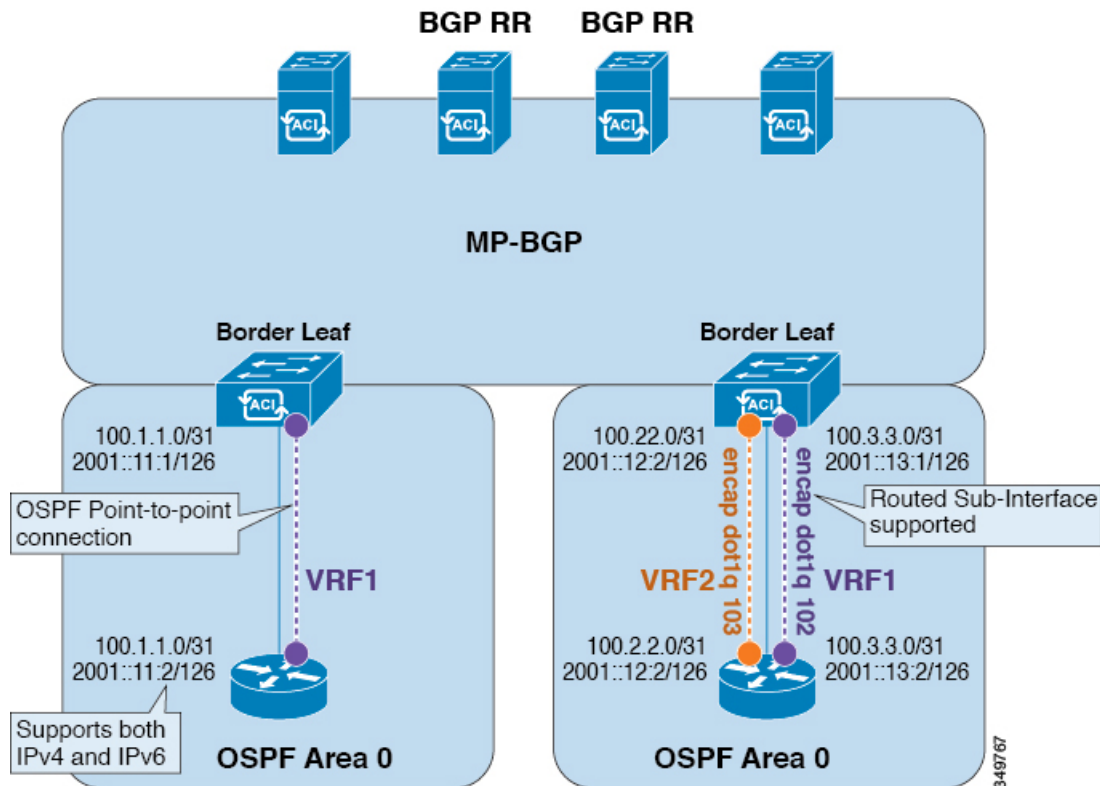
OSPF

OSPF Layer 3 Outside Connections

OSPF Layer 3 Outside connections can be normal or NSSA areas. The backbone (area 0) area is also supported as an OSPF Layer 3 Outside connection area. ACI supports both OSPFv2 for IPv4 and OSPFv3 for IPv6. When creating an OSPF Layer 3 Outside, it is not necessary to configure the OSPF version. The correct OSPF process is created automatically based on the interface profile configuration (IPv4 or IPv6 addressing). Both IPv4 and IPv6 protocols are supported on the same interface (dual stack) but it is necessary to create two separate interface profiles.

Layer 3 Outside connections are supported for the routed interfaces, routed sub-interfaces, and SVIs. The SVIs are used when there is a need to share the physical connect for both Layer 2 and Layer 3 traffic. The SVIs are supported on ports, port channels, and virtual port channels (vPCs).

Figure 49: OSPF Layer3 Out Connections



When an SVI is used for an Layer 3 Outside connection, an external bridge domain is created on the border leaf switches. The external bridge domain allows connectivity between the two VPC switches across the ACI fabric. This allows both the VPC switches to establish the OSPF adjacencies with each other and the external OSPF device.

When running OSPF over a broadcast network, the time to detect a failed neighbor is the dead time interval (default 40 seconds). Reestablishing the neighbor adjacencies after a failure may also take longer due to designated router (DR) election.

**Note**

- A link or port channel failure to one vPC Node does not cause an OSPF adjacency to go down. The OSPF adjacency can stay up using the external bridge domain accessible through the other vPC node.
- When an OSPF time policy or a BGP, OSPF, or EIGRP address family policy is applied to an L3Out, you can observe the following behaviors:
 - If the L3Out and the policy are defined in the same tenant, then there is no change in behavior.
 - If the L3Out is configured in a user tenant other than the common tenant, the L3Out VRF instance is resolved to the common tenant, and the policy is defined in the common tenant, then only the default values are applied. Any change in the policy will not take effect.
- If a border leaf switch forms OSPF adjacency with two external switches and one of the two switches experiences a route loss while the adjacent switches does not, the Cisco ACI border leaf switch reconverges the route for both neighbors.
- OSPF supports aggressive timers. However, these timers quickly bring down the adjacency and cause CPU churn. Therefore, we recommend that you use the default timers and use bidirectional forwarding detection (BFD) to get sub-second failure detection.

Creating OSPF External Routed Network for Management Tenant Using REST API

- You must verify that the router ID and the logical interface profile IP address are different and do not overlap.
- The following steps are for creating an OSPF external routed network for a management tenant. To create an OSPF external routed network for a tenant, you must choose a tenant and create a VRF for the tenant.
- For more details, see *Cisco APIC and Transit Routing*.

Create an OSPF external routed network for management tenant.

Example:

POST: <https://apic-ip-address/api/mo/uni/tn-mgmt.xml>

```
<fvTenant name="mgmt">
  <fvBD name="bd1">
    <fvRsBDToOut tnL3extOutName="RtdOut" />
    <fvSubnet ip="1.1.1.1/16" />
    <fvSubnet ip="1.2.1.1/16" />
    <fvSubnet ip="40.1.1.1/24" scope="public" />
    <fvRsCtx tnFvCtxName="inb" />
  </fvBD>
</fvTenant>
<fvCtx name="inb" />

<l3extOut name="RtdOut">
  <l3extRsL3DomAtt tDn="uni/l3dom-extdom"/>
  <l3extInstP name="extMgmt">
    </l3extInstP>
  <l3extLNodeP name="borderLeaf">
    <l3extRsNodeL3OutAtt tDn="topology/pod-1/node-101" rtrId="10.10.10.10"/>
    <l3extRsNodeL3OutAtt tDn="topology/pod-1/node-102" rtrId="10.10.10.11"/>
    <l3extLIIfP name='portProfile'>

```

```

        <l3extRsPathL3OutAtt tDn="topology/pod-1/paths-101/pathep-[eth1/40]" ifInstT='l3-port'
addr="192.168.62.1/24"/>
        <l3extRsPathL3OutAtt tDn="topology/pod-1/paths-102/pathep-[eth1/40]" ifInstT='l3-port'
addr="192.168.62.5/24"/>
        <ospfIfP/>
                                </l3extLIIfP>
    </l3extLNodeP>
    <l3extRsEctx tnFvCtxName="inb"/>
    <ospfExtP areaId="57" />
</l3extOut>
</fvTenant>

```

EIGRP

Overview

This article provides a typical example of how to configure Enhanced Interior Gateway Routing Protocol (EIGRP) when using the Cisco APIC. The following information applies when configuring EIGRP:

- The tenant, VRF, and bridge domain must already be created.
- The Layer 3 outside tenant network must already be configured.
- The route control profile under routed outside must already be configured.
- The EIGRP VRF policy is the same as the EIGRP family context policy.
- EIGRP supports only export route control profile. The configuration related to route controls is common across all the protocols.

You can configure EIGRP to perform automatic summarization of subnet routes (route summarization) into network-level routes. For example, you can configure subnet 131.108.1.0 to be advertised as 131.108.0.0 over interfaces that have subnets of 192.31.7.0 configured. Automatic summarization is performed when there are two or more network router configuration commands configured for the EIGRP process. By default, this feature is enabled.

For more information about route summarization, see the *Cisco Application Centric Infrastructure Fundamentals Guide*.

Configuring EIGRP Using the REST API

Step 1 Configure an EIGRP context policy.

Example:

```

<polUni>
  <fvTenant name="cisco_6">
    <eigrpCtxAfPol actIntvl="3" descr="" dn="uni/tn-cisco_6/eigrpCtxAfPol-eigrp_default_pol"
extDist="170"
    intDist="90" maxPaths="8" metricStyle="narrow" name="eigrp_default_pol" ownerKey=""
ownerTag="" />
  </fvTenant>
</polUni>

```

Step 2 Configure an EIGRP interface policy.

Example:

```
<polUni>
  <fvTenant name="cisco_6">
    <eigrpIfPol bw="10" ctrl="nh-self,split-horizon" delay="10" delayUnit="tens-of-micro" descr=""
dn="uni/tn-cisco_6/eigrpIfPol-eigrp_if_default"
    helloIntvl="5" holdIntvl="15" name="eigrp_if_default" ownerKey="" ownerTag=""/>
  </fvTenant>
</polUni>
```

Step 3 Configure an EIGRP VRF.**Example:**

IPv4:

```
<polUni>
  <fvTenant name="cisco_6">
    <fvCtx name="dev">
      <fvRsCtxToEigrpCtxAfPol tnEigrpCtxAfPolName="eigrp_ctx_pol_v4" af="1"/>
    </fvCtx>
  </fvTenant>
</polUni>
```

IPv6:

```
<polUni>
  <fvTenant name="cisco_6">
    <fvCtx name="dev">
      <fvRsCtxToEigrpCtxAfPol tnEigrpCtxAfPolName="eigrp_ctx_pol_v6" af="ipv6-ucast"/>
    </fvCtx>
  </fvTenant>
</polUni>
```

Step 4 Configure an EIGRP Layer3 Outside.**Example:**

IPv4

```
<polUni>
  <fvTenant name="cisco_6">
    <l3extOut name="ext">
      <eigrpExtP asn="4001"/>
      <l3extLNodeP name="node1">
        <l3extLIfP name="intf_v4">
          <l3extRsPathL3OutAtt addr="201.1.1.1/24" ifInstT="l3-port"
tDn="topology/pod-1/paths-101/pathep-[eth1/4]"/>
          <eigrpIfP name="eigrp_ifp_v4">
            <eigrpRsIfPol tnEigrpIfPolName="eigrp_if_pol_v4"/>
          </eigrpIfP>
        </l3extLIfP>
      </l3extLNodeP>
    </l3extOut>
  </fvTenant>
</polUni>
```

IPv6

```
<polUni>
  <fvTenant name="cisco_6">
    <l3extOut name="ext">
      <eigrpExtP asn="4001"/>
      <l3extLNodeP name="node1">
        <l3extLIfP name="intf_v6">
          <l3extRsPathL3OutAtt addr="2001::1/64" ifInstT="l3-port"
tDn="topology/pod-1/paths-101/pathep-[eth1/4]"/>
        </l3extLIfP>
      </l3extLNodeP>
    </l3extOut>
  </fvTenant>
</polUni>
```

```

        <eigrpIfP name="eigrp_ifp_v6">
          <eigrpRsIfPol tnEigrpIfPolName="eigrp_if_pol_v6"/>
        </eigrpIfP>
      </l3extLIfP>
    </l3extLNodeP>
  </l3extOut>
</fvTenant>
</polUni>

```

IPv4 and IPv6

```

<polUni>
  <fvTenant name="cisco_6">
    <l3extOut name="ext">
      <eigrpExtP asn="4001"/>
      <l3extLNodeP name="node1">
        <l3extLIfP name="intf_v4">
          <l3extRsPathL3OutAtt addr="201.1.1.1/24" ifInstT="l3-port"
            tDn="topology/pod-1/paths-101/pathep-[eth1/4]"/>
          <eigrpIfP name="eigrp_ifp_v4">
            <eigrpRsIfPol tnEigrpIfPolName="eigrp_if_pol_v4"/>
          </eigrpIfP>
        </l3extLIfP>

        <l3extLIfP name="intf_v6">
          <l3extRsPathL3OutAtt addr="2001::1/64" ifInstT="l3-port"
            tDn="topology/pod-1/paths-101/pathep-[eth1/4]"/>
          <eigrpIfP name="eigrp_ifp_v6">
            <eigrpRsIfPol tnEigrpIfPolName="eigrp_if_pol_v6"/>
          </eigrpIfP>
        </l3extLIfP>
      </l3extLNodeP>
    </l3extOut>
  </fvTenant>
</polUni>

```

Step 5 (Optional) Configure the interface policy knobs.

Example:

```

<polUni>
  <fvTenant name="cisco_6">
    <eigrpIfPol bw="1000000" ctrl="nh-self,split-horizon" delay="10"
      delayUnit="tens-of-micro" helloIntvl="5" holdIntvl="15" name="default"/>
  </fvTenant>
</polUni>

```

The `bandwidth (bw)` attribute is defined in Kbps. The `delayUnit` attribute can be "tens of micro" or "pico".

Neighbor Discovery

Neighbor Discovery

The IPv6 Neighbor Discovery (ND) protocol is responsible for the address auto configuration of nodes, discovery of other nodes on the link, determining the link-layer addresses of other nodes, duplicate address detection, finding available routers and DNS servers, address prefix discovery, and maintaining reachability information about the paths to other active neighbor nodes.

ND-specific Neighbor Solicitation or Neighbor Advertisement (NS or NA) and Router Solicitation or Router Advertisement (RS or RA) packet types are supported on all ACI fabric Layer 3 interfaces, including physical, Layer 3 sub interface, and SVI (external and pervasive). Up to APIC release 3.1(1x), RS/RA packets are used for auto configuration for all Layer 3 interfaces but are only configurable for pervasive SVIs.

Starting with APIC release 3.1(2x), RS/RA packets are used for auto configuration and are configurable on Layer 3 interfaces including routed interface, Layer 3 sub interface, and SVI (external and pervasive).

ACI bridge domain ND always operates in flood mode; unicast mode is not supported.

The ACI fabric ND support includes the following:

- Interface policies (`nd:IfPol`) control ND timers and behavior for NS/NA messages.
- ND prefix policies (`nd:PxPol`) control RA messages.
- Configuration of IPv6 subnets for ND (`fv:Subnet`).
- ND interface policies for external networks.
- Configurable ND subnets for external networks, and arbitrary subnet configurations for pervasive bridge domains are not supported.

Configuration options include the following:

- Adjacencies
 - Configurable Static Adjacencies: (<vrf, L3Iface, ipv6 address> --> mac address)
 - Dynamic Adjacencies: Learned via exchange of NS/NA packets
- Per Interface
 - Control of ND packets (NS/NA)
 - Neighbor Solicitation Interval
 - Neighbor Solicitation Retry count
 - Control of RA packets
 - Suppress RA
 - Suppress RA MTU
 - RA Interval, RA Interval minimum, Retransmit time
- Per Prefix (advertised in RAs) control
 - Lifetime, preferred lifetime
 - Prefix Control (auto configuration, on link)
- Neighbor Discovery Duplicate Address Detection (DAD)

Creating the Tenant, VRF, and Bridge Domain with IPv6 Neighbor Discovery on the Bridge Domain Using the REST API

Create a tenant, VRF, bridge domain with a neighbor discovery interface policy and a neighbor discovery prefix policy.

Example:

```
<fvTenant descr="" dn="uni/tn-ExampleCorp" name="ExampleCorp" ownerKey="" ownerTag="">
  <ndIfPol name="NDPol001" ctrl="managed-cfg" descr="" hopLimit="64" mtu="1500" nsIntvl="1000"
nsRetries="3" ownerKey="" ownerTag="" raIntvl="600" raLifetime="1800" reachableTime="0"
retransTimer="0"/>
  <fvCtx descr="" knwMcastAct="permit" name="pvnl" ownerKey="" ownerTag="" pcEnfPref="enforced">
    </fvCtx>
  <fvBD arpFlood="no" descr="" mac="00:22:BD:F8:19:FF" multiDstPktAct="bd-flood" name="bd1"
ownerKey="" ownerTag="" unicastRoute="yes" unkMacUcastAct="proxy" unkMcastAct="flood">
  <fvRsBDToNDP tnNdIfPolName="NDPol001"/>
  <fvRsCtx tnFvCtxName="pvnl"/>
  <fvSubnet ctrl="nd" descr="" ip="34::1/64" name="" preferred="no" scope="private">
    <fvRsNdPfxPol tnNdPfxPolName="NDPfxPol001"/>
  </fvSubnet>
  <fvSubnet ctrl="nd" descr="" ip="33::1/64" name="" preferred="no" scope="private">
    <fvRsNdPfxPol tnNdPfxPolName="NDPfxPol002"/>
  </fvSubnet>
</fvBD>
  <ndPfxPol ctrl="auto-cfg,on-link" descr="" lifetime="1000" name="NDPfxPol001" ownerKey=""
ownerTag="" prefLifetime="1000"/>
  <ndPfxPol ctrl="auto-cfg,on-link" descr="" lifetime="4294967295" name="NDPfxPol002" ownerKey=""
ownerTag="" prefLifetime="4294967295"/>
</fvTenant>
```

Note If you have a public subnet when you configure the routed outside, you must associate the bridge domain with the outside configuration.

Guidelines and Limitations

The following guidelines and limitations apply to Neighbor Discovery Router Advertisement (ND RA) Prefixes for Layer 3 Interfaces:

- An ND RA configuration applies only to IPv6 Prefixes. Any attempt to configure an ND policy on IPv4 Prefixes will fail to apply.

Configuring an IPv6 Neighbor Discovery Interface Policy with RA on a Layer 3 Interface Using the REST API

Configure an IPv6 neighbor discovery interface policy and associate it with a Layer 3 interface:

The following example displays the configuration in a non-VPC set up.

Example:

```

<fvTenant dn="uni/tn-ExampleCorp" name="ExampleCorp">
  <ndIfPol name="NDPol001" ctrl="managed-cfg" hopLimit="64" mtu="1500" nsIntvl="1000" nsRetries="3"
  raIntvl="600" raLifetime="1800" reachableTime="0" retransTimer="0"/>
  <fvCtx name="pvn1" pcEnfPref="enforced">
    </fvCtx>
  <l3extOut enforceRtctrl="export" name="l3extOut001">
    <l3extRsEctx tnFvCtxName="pvn1"/>
    <l3extLNodeP name="lnodeP001">
      <l3extRsNodeL3OutAtt rtrId="11.11.205.1" rtrIdLoopBack="yes" tDn="topology/pod-2/node-2011"/>
      <l3extLIIfP name="lifP001">
        <l3extRsPathL3OutAtt addr="2001:20:21:22::2/64" ifInstT="l3-port" llAddr="::"
        mac="00:22:BD:F8:19:FF" mode="regular" mtu="inherit" tDn="topology/pod-2/paths-2011/pathep-[eth1/1]">
          <ndPfxP>
            <ndRsPfxPToNdPfxPol tnNdPfxPolName="NDPfxPol001"/>
          </ndPfxP>
        </l3extRsPathL3OutAtt>
        <l3extRsNdIfPol tnNdIfPolName="NDPol001"/>
      </l3extLIIfP>
    </l3extLNodeP>
    <l3extInstP name="instp"/>
  </l3extOut>
  <ndPfxPol ctrl="auto-cfg,on-link" descr="" lifetime="1000" name="NDPfxPol001" ownerKey="" ownerTag=""
  prefLifetime="1000"/>
</fvTenant>

```

Note For VPC ports, ndPfxP must be a child of l3extMember instead of l3extRsNodeL3OutAtt. The following code snippet shows the configuration in a VPC setup.

```

<l3extLNodeP name="lnodeP001">
<l3extRsNodeL3OutAtt rtrId="11.11.205.1" rtrIdLoopBack="yes" tDn="topology/pod-2/node-2011"/>
<l3extRsNodeL3OutAtt rtrId="12.12.205.1" rtrIdLoopBack="yes" tDn="topology/pod-2/node-2012"/>
  <l3extLIIfP name="lifP002">
    <l3extRsPathL3OutAtt addr="0.0.0.0" encap="vlan-205" ifInstT="ext-svi" llAddr="::"
    mac="00:22:BD:F8:19:FF" mode="regular" mtu="inherit"
    tDn="topology/pod-2/protpaths-2011-2012/pathep-[vpc7]" >
      <l3extMember addr="2001:20:25:1::1/64" descr="" llAddr="::" name="" nameAlias="" side="A">
        <ndPfxP >
          <ndRsPfxPToNdPfxPol tnNdPfxPolName="NDPfxPol001"/>
        </ndPfxP>
      </l3extMember>
      <l3extMember addr="2001:20:25:1::2/64" descr="" llAddr="::" name="" nameAlias="" side="B">
        <ndPfxP >
          <ndRsPfxPToNdPfxPol tnNdPfxPolName="NDPfxPol001"/>
        </ndPfxP>
      </l3extMember>
    </l3extRsPathL3OutAtt>
  </l3extLIIfP>
</l3extLNodeP>

```

Microsoft NLB

Configuring Microsoft NLB in Unicast Mode Using the REST API

To configure Microsoft NLB in unicast mode, send a post with XML such as the following example:

Example:

`https://apic-ip-address/api/node/mo/uni/.xml`

```
<polUni>
  <fvTenant name="tn2" >
    <fvCtx name="ctx1"/>
    <fvBD name="bd2">
      <fvRsCtx tnFvCtxName="ctx1" />
    </fvBD>
    <fvAp name = "ap1">
      <fvAEPg name = "ep1">
        <fvRsBd tnFvBDName = "bd2"/>
        <fvSubnet ip="10.0.1.1/32" scope="public" ctrl="no-default-gateway">
          <fvEpNlb mac="12:21:21:35" mode="mode-uc"/>
        </fvSubnet>
      </fvAEPg>
    </fvAp>
  </fvTenant>
</polUni>
```

Configuring Microsoft NLB in Multicast Mode Using the REST API

To configure Microsoft NLB in multicast mode, send a post with XML such as the following example:

Example:

`https://apic-ip-address/api/node/mo/uni/.xml`

```
<polUni>
  <fvTenant name="tn2" >
    <fvCtx name="ctx1"/>
    <fvBD name="bd2">
      <fvRsCtx tnFvCtxName="ctx1" />
    </fvBD>
    <fvAp name = "ap1">
      <fvAEPg name = "ep1">
        <fvRsBd tnFvBDName = "bd2"/>
        <fvSubnet ip="2001:0db8:85a3:0000:0000:8a2e:0370:7344/128" scope="public"
ctrl="no-default-gateway">
          <fvEpNlb mac="03:21:21:35" mode="mode-mcast--static"/>
        </fvSubnet>
        <fvRsPathAtt tDn="topology/pod-1/paths-101/pathep-[eth1/6]" encap="vlan-911" >
          <fvNlbStaticGroup mac = "03:21:21:35" />
        </fvRsPathAtt>
      </fvAEPg>
    </fvAp>
  </fvTenant>
```

```
</polUni>
```

Configuring Microsoft NLB in IGMP Mode Using the REST API

To configure Microsoft NLB in IGMP mode, send a post with XML such as the following example:

Example:

```
https://apic-ip-address/api/node/mo/uni/.xml
```

```
<polUni>
  <fvTenant name="tn2" >
    <fvCtx name="ctx1"/>
    <fvBD name="bd2">
      <fvRsCtx tnFvCtxName="ctx1" />
    </fvBD>
    <fvAp name = "ap1">
      <fvAEPg name = "ep1">
        <fvRsBd tnFvBDName = "bd2"/>
        <fvSubnet ip="10.0.1.3/32" scope="public" ctrl="no-default-gateway">
          <fvEpNlb group = "224.132.18.17" mode="mode-mcast-igmp" />
        </fvSubnet>
      </fvAEPg>
    </fvAp>
  </fvTenant>
</polUni>
```

MLD Snooping

Configuring and Assigning an MLD Snooping Policy to a Bridge Domain using the REST API

To configure an MLD Snooping policy and assign it to a bridge domain, send a post with XML such as the following example:

Example:

```
https://apic-ip-address/api/node/mo/uni/.xml
```

```
<fvTenant name="mldsn">
  <mldSnoopPol adminSt="enabled" ctrl="fast-leave,querier" name="mldsn-it-fabric-querier-policy"
  queryIntvl="125"
  rspIntvl="10" startQueryCnt="2" startQueryIntvl="31" status="" />
  <fvBD name="mldsn-bd3">
    <fvRsMldsn status="" tnMldSnoopPolName="mldsn-it-policy"/>
  </fvBD>
</fvTenant>
```

This example creates and configures the MLD Snooping policy `mldsn` with the following properties, and binds the MLD policy `mldsn-it-fabric-querier-policy` to bridge domain `mldsn-bd3`:

- Fast leave processing is enabled
 - Querier processing is enabled
 - Query Interval is set at 125
 - Max query response time is set at 10
 - Number of initial queries to send is set at 2
 - Time for sending initial queries is set at 31
-



CHAPTER 16

Configuring QoS

- [QoS for L3Outs, on page 405](#)
- [CoS Preservation, on page 407](#)
- [Multipod QoS, on page 408](#)
- [Translating QoS Ingress Markings to Egress Markings, on page 410](#)

QoS for L3Outs

L3Outs QoS

L3Out QoS can be configured using Contracts applied at the external EPG level. Starting with Release 4.0(1), L3Out QoS can also be configured directly on the L3Out interfaces.



Note If you are running Cisco APIC Release 4.0(1) or later, we recommend using the custom QoS policies applied directly to the L3Out to configure QoS for L3Outs.

Packets are classified using the ingress DSCP or CoS value so it is possible to use custom QoS policies to classify the incoming traffic into Cisco ACI QoS queues. A custom QoS policy contains a table mapping the DSCP/CoS values to the user queue and to the new DSCP/CoS value (in case of marking). If there is no mapping for a specific DSCP/CoS value, the user queue is selected by the QoS priority setting of the ingress L3Out interface if configured.

Configuring QoS Directly on L3Out Using REST API

This section describes how to configure QoS directly on an L3Out. This is the preferred way of configuring L3Out QoS starting with Cisco APIC Release 4.0(1).

You can configure QoS for L3Out on one of the following objects:

- Switch Virtual Interface (SVI)
- Sub Interface
- Routed Outside

Step 1 Configure QoS priorities for a L3Out SVI.**Example:**

```
<l3extLlIfP descr="" dn="uni/tn-DT/out-L3_4_2_24_SVI17/lnodep-L3_4_E2_24/lifp-L3_4_E2_24_SVI_19"
  name="L3_4_E2_24_SVI_19" prio="level6" tag="yellow-green">
  <l3extRsPathL3OutAtt addr="0.0.0.0" autostate="disabled" descr="SVI19" encap="vlan-19"
    encapScope="local" ifInstT="ext-svi" ipv6Dad="enabled" llAddr="::"
    mac="00:22:BD:F8:19:FF" mode="regular" mtu="inherit"
    tDn="topology/pod-1/protpaths-103-104/pathep-[V_L3_14_2-24]"
    targetDscp="unspecified">
    <l3extMember addr="107.2.1.253/24" ipv6Dad="enabled" llAddr="::" side="B"/>
    <l3extMember addr="107.2.1.252/24" ipv6Dad="enabled" llAddr="::" side="A"/>
  </l3extRsPathL3OutAtt>
  <l3extRsLlIfPCustQosPol tnQosCustomPolName="VrfQos006"/>
</l3extLlIfP>
```

Step 2 Configure QoS priorities for a sub-interface.**Example:**

```
<l3extLlIfP dn="uni/tn-DT/out-L4E48_inter_tenant/lnodep-L4E48_inter_tenant/lifp-L4E48"
  name="L4E48" prio="level4" tag="yellow-green">
  <l3extRsPathL3OutAtt addr="210.1.0.254/16" autostate="disabled" encap="vlan-20"
    encapScope="local" ifInstT="sub-interface" ipv6Dad="enabled" llAddr="::"
    mac="00:22:BD:F8:19:FF" mode="regular" mtu="inherit"
    tDn="topology/pod-1/paths-104/pathep-[eth1/48]" targetDscp="unspecified"/>
  <l3extRsNdIfPol annotation="" tnNdIfPolName=""/>
  <l3extRsLlIfPCustQosPol annotation="" tnQosCustomPolName=" vrfQos002"/>
</l3extLlIfP>
```

Step 3 Configure QoS priorities for a routed outside.**Example:**

```
<l3extLlIfP dn="uni/tn-DT/out-L2E37/lnodep-L2E37/lifp-L2E37OUT"
  name="L2E37OUT" prio="level5" tag="yellow-green">
  <l3extRsPathL3OutAtt addr="30.1.1.1/24" autostate="disabled" encap="unknown"
    encapScope="local" ifInstT="l3-port" ipv6Dad="enabled"
    llAddr="::" mac="00:22:BD:F8:19:FF" mode="regular"
    mtu="inherit" targetDscp="unspecified"
    tDn="topology/pod-1/paths-102/pathep-[eth1/37]"/>
  <l3extRsNdIfPol annotation="" tnNdIfPolName=""/>
  <l3extRsLlIfPCustQosPol tnQosCustomPolName="vrfQos002"/>
</l3extLlIfP>
```

Configuring QoS Contract for L3Out Using REST API

This section describes how to configure QoS for L3Outs using Contracts.



Note Starting with Release 4.0(1), we recommend using custom QoS policies for L3Out QoS as described in [Configuring QoS Directly on L3Out Using REST API, on page 405](#) instead.

Step 1 When configuring the tenant, VRF, and bridge domain, configure the VRF for egress mode (`pcEnfDir="egress"`) with policy enforcement enabled (`pcEnfPref="enforced"`). Send a post with XML similar to the following example:

Example:

```
<fvTenant name="t1">
  <fvCtx name="v1" pcEnfPref="enforced" pcEnfDir="egress"/>
  <fvBD name="bd1">
    <fvRsCtx tnFvCtxName="v1"/>
    <fvSubnet ip="44.44.44.1/24" scope="public"/>
    <fvRsBDToOut tnL3extOutName="l3out1"/>
  </fvBD>/>
</fvTenant>
```

Step 2 When creating the filters and contracts to enable the EPGs participating in the L3Out to communicate, configure the QoS priority.

The contract in this example includes the QoS priority, `level1`, for traffic ingressing on the L3Out. Alternatively, it could define a target DSCP value. QoS policies are supported on either the contract or the subject.

The filter also has the `matchDscp="EF"` criteria, so that traffic with this specific TAG received by the L3out processes through the queue specified in the contract subject.

Note VRF enforcement should be ingress, for QoS or custom QoS on L3out interface, VRF enforcement need be egress, only when the QoS classification is going to be done in the contract for traffic between EPG and L3out or L3out to L3out.

Note If QoS classification is set in the contract and VRF enforcement is egress, then contract QoS classification would override the L3out interface QoS or Custom QoS classification, So either we need to configure this one or the new one.

Example:

```
<vzFilter name="http-filter">
  <vzEntry name="http-e" etherT="ip" prot="tcp" matchDscp="EF"/>
</vzFilter>
<vzBrCP name="httpCtrct" prio="level1" scope="context">
  <vzSubj name="subj1">
    <vzRsSubjFiltAtt tnVzFilterName="http-filter"/>
  </vzSubj>
</vzBrCP>
```

CoS Preservation

Class of Service (CoS) Preservation for Ingress and Egress Traffic

When traffic enters the Cisco ACI fabric, each packet's priority is mapped to a Cisco ACI QoS level. These QoS levels are then stored in the CoS field and DE bit of the packet's outer header while the original headers are discarded.

If you want to preserve the original CoS values of the ingressing packets and restore it when the packet leaves the fabric, you can enable the 802.1p Class of Service (CoS) preservation using a global fabric QoS policy as described in this section.

The CoS preservation is supported in single pod and multipod topologies, however in multipod topologies, CoS preservation can be used only when you are not concerned with preserving the settings in the IPN between pods. To preserve the CoS values of the packets as they are transiting the IPN, use the DSCP translation policy as described in [Multipod QoS and DSCP Translation Policy, on page 408](#).

Enable Class Of Service (CoS) Preservation Using REST API

This section describes how to enable CoS preservation to ensure that QoS priority settings are handled the same for traffic entering and transiting a single-pod fabric as for traffic entering one pod and egressing another in a multipod fabric.



Note Enabling CoS preservation applies a default CoS-to-DSCP mapping to the various traffic types.

Enable CoS preservation.

```
POST https://<apic-ip>/api/node/mo/uni/infra/qosinst-default.xml
```

Example:

```
<qosInstPol name="default" dn="uni/infra/qosinst-default" ctrl="dot1p-preserve"/>
```

Disable CoS preservation.

Example:

```
<qosInstPol name="default" dn="uni/infra/qosinst-default" ctrl=""/>
```

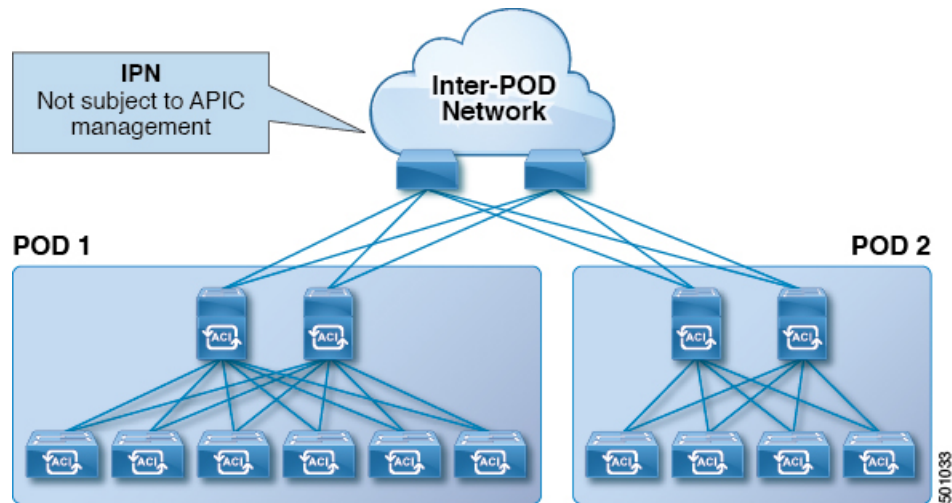
Multipod QoS

Multipod QoS and DSCP Translation Policy

When traffic is sent and received within the Cisco ACI fabric, the QoS Level is determined based on the CoS value of the VXLAN packet's outer header. In multipod topologies, where devices that are not under Cisco APIC's management may modify the CoS values in the transiting packets, you can preserve the QoS Level setting by creating a mapping between the Cisco ACI and the DSCP value within the packet.

If you are not concerned with preserving the QoS settings in the IPN traffic between pods, but would like to preserve the original CoS values of the packets ingressing and egressing the fabric, see [Class of Service \(CoS\) Preservation for Ingress and Egress Traffic, on page 407](#) instead.

Figure 50: Multipod Topology



As illustrated in this figure, traffic between pods in a multipod topology passes through an IPN, which may contain devices that are not under Cisco APIC's management. When a network packet is sent from a spine or a leaf switch in POD1, the devices in the IPN may modify the 802.1p value in the packet. In this case, when the frame reaches a spine or a leaf switch in POD2, it would have an 802.1p value that was assigned by the IPN device, instead of the Cisco ACI QoS Level value assigned at the source in POD1.

In order to preserve the proper QoS Level of the packet and avoid high priority packets from being delayed or dropped, you can use a DSCP translation policy for traffic that goes between multiple PODs connected by an IPN. When a DSCP translation policy is enabled, Cisco APIC converts the QoS Level value (represented by the CoS value of the VXLAN packet) to a DSCP value according to the mapping rules you specify. When a packet sent from POD1 reaches POD2, the mapped DSCP value is translated back into the original CoS value for the appropriate QoS Level.

Creating DSCP Translation Policy Using REST API

This section describes how to create a DSCP translation policy to guarantee QoS Level settings across multiple PODs connected by an IPN.

Step 1 Enable and configure a DSCP translation policy.

```
POST https://<apic-ip>/api/node/mo/uni/tn-infra/dscptranspol-default.xml
```

Example:

```
<qosDscpTransPol dn="uni/tn-infra/dscptranspol-default" adminSt="enabled"
  traceroute="AF43" span="AF42" policy="AF22" level3="AF13"
  level2="AF12" level1="AF11" control="AF21" />
```

Step 2 Disable the DSCP translation policy.

```
POST https://<apic-ip>/api/node/mo/uni/tn-infra/dscptranspol-default.xml
```

Example:

```
<qosDscpTransPol dn="uni/tn-infra/dscptranspol-default" adminSt="disabled"
  traceroute="AF43" span="AF42" policy="AF22" level3="AF13"
  level2="AF12" level1="AF11" control="AF21"/>
```

Translating QoS Ingress Markings to Egress Markings

Translating Ingress to Egress QoS Markings

Cisco APIC enables translating the DSCP and CoS values of the ingressing traffic to a QoS Level to be used inside the Cisco ACI fabric. Translation is supported only if the DSCP values are present in the IP packet and CoS values are present in the Ethernet frames.

For example, this functionality allows the Cisco ACI fabric to classify the traffic for devices that classify the traffic based only on the CoS value, such as Layer-2 packets, which do not have an IP header.

CoS Translation Guidelines and Limitations

You must enable the global fabric CoS preservation policy, as described in [Class of Service \(CoS\) Preservation for Ingress and Egress Traffic, on page 407](#).

CoS translation is not supported on external L3 interfaces.

CoS translation is supported only if the egress frame is 802.1Q encapsulated.

CoS translation is not supported when the following configuration options are enabled:

- Contracts are configured that include QoS.
- The outgoing interface is on a FEX.
- Multipod QoS using a DSCP policy is enabled.
- Dynamic packet prioritization is enabled.
- If an EPG is configured with intra-EPG endpoint isolation enforced.
- If an EPG is configured with allow-microsegmentation enabled.

Creating Custom QoS Policy Using REST API

This section describes how to create a custom QoS policy and associate it with an EPG using the REST API.

Before you begin

You must have created the tenant, application, and EPGs that will consume the custom QoS policy.

Step 1 Create a custom QoS policy.

Example:

```
<qosCustomPol name="vrfQos001" dn="uni/tn-t001/qoscustom-vrfQos001">
  <qosDscpClass to="AF31" targetCos="6" target="unspecified">
```

```

        prio="unspecified" from="AF23"/>
    <qosDot1PClass to="1" targetCos="6" target="unspecified"
        prio="unspecified" from="0"/>
</qosCustomPol>

```

Step 2 Associate the policy with an EPG that will consume it.

Example:

```

<fvAEPg prio="unspecified" prefGrMemb="exclude" pcEnfPref="unenforced"
    name="ep2" matchT="AtleastOne" isAttrBasedEPg="no" fwdCtrl=""
    dn="uni/tn-t001/ap-ap2/epg-ep2">
    <fvRsDomAtt tDn="uni/vmmp-VMware/dom-vs1" resImedcy="lazy"
        primaryEncap="unknown" netflowPref="disabled"
        instrImedcy="lazy" encapMode="auto" encap="unknown"
        delimiter="" classPref="encap"/>
    <fvRsCustQosPol tnQosCustomPolName="vrfQos001"/>
    <fvRsBd tnFvBDName="default"/>
</fvAEPg>

```

Troubleshooting Cisco APIC QoS Policies

The following table summarizes common troubleshooting scenarios for Cisco APIC QoS.

Problem	Solution
Unable to update a configured QoS policy.	<ol style="list-style-type: none"> Invoke the following API to ensure that <code>qospDscpRule</code> is present on the leaf. <pre>GET https://192.0.20.123/api/node/class/qospDscpRule.xml</pre> Ensure that the QoS rules are accurately configured and associated to the EPG ID to which the policy is attached. Use the following NX-OS style CLI commands to verify the configuration. <pre>leaf1# show vlan leaf1# show system internal aclqos qos policy detail apic1# show running-config tenant <i>tenant-name</i> policy-map type qos <i>custom-qos-policy-name</i> apic1# show running-config tenant <i>tenant-name</i> application <i>application-name</i> epg <i>epg-name</i></pre>

Problem	Solution
Show QoS interface statistics.	<p>CLI displays statistics for eth1/1 for only QoS classes – level1, leve2, level3, level4, level5, level6, and policy-plane – if you don't use “detail” option.</p> <pre>NXOS ibash cli: tor-leaf1# show queuing interface ethernet 1/1 [detail]</pre> <p>If you want to display statistics for control-plane and span classes for an interface, you need to use CLI with the “detail” option.</p> <p>Example: fabric 107 show queuing interface ethernet 1/1 detail</p> <pre>APIC CLI: swtb123-ifc1# fabric node_id show queuing interface ethernet 1/1</pre>



CHAPTER 17

Managing Layer 4 to Layer 7 Services

- [About Layer 4 to Layer 7 Services, on page 413](#)
- [Access for Managing Layer 4 to Layer 7 Services, on page 414](#)
- [Device Packages, on page 417](#)
- [Trunking, on page 419](#)
- [Device Selection Policies, on page 420](#)
- [Policy Based Redirect and Service Nodes Tracking, on page 421](#)
- [Service Graph Templates, on page 427](#)
- [Layer 4 to Layer 7 Parameters, on page 429](#)
- [Copy Services, on page 433](#)
- [Developing Automation, on page 435](#)
- [Example: Configuring Layer 4 to Layer 7 Services \(Firewall\), on page 443](#)
- [Example: Configuring Layer 4 to Layer 7 Route Peering, on page 452](#)

About Layer 4 to Layer 7 Services

About Application-Centric Infrastructure Layer 4 to Layer 7 Services

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

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

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

For a VMM domain, configure VMM domain credentials and configure a vCenter/vShield controller profile.

- A VLAN pool with an encapsulation block range
- At least one contract
- At least one EPG

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

1. Import a Device Package .

Only the provider administrator can import the device package.

2. Register the device and the logical interfaces.

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

3. Create a Logical Device.

4. Configure device parameters.

5. Optional. If you are configuring an ASA Firewall service, enable trunking on the device.

6. Configure a Device Selection Policy.

7. Configure a Service Graph Template.

- a. Select the default service graph template parameters from an application profile.
- b. Configure additional service graph template parameters, if needed.

8. Attach the service graph template to a contract.

9. Configure additional configuration parameters, if needed.

For more information about deploying Layer 4 to Layer 7 services, see the *Cisco APIC Layer 4 to Layer 7 Services Deployment Guide*.

Access for Managing Layer 4 to Layer 7 Services

Configure In-Band Connectivity to Devices Using Tenant's VRF Using the REST API

The following is an example of using REST APIs to configure in-band connectivity to devices 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="tenant1">
    <fvCtx name="mgmt_ctx1"/>
    <vnsCtrlrMgmtPol ctxDn="uni/tn-tenant1/ctx-mgmt_ctx1">
      <vnsRsMgmtAddr tDn="uni/tn-tenant1/ap-services/epg-ifc/CtrlrAddrInst-ifc"/>
    </vnsCtrlrMgmtPol>
    <fvBD name="mgmt_ServicesMgmtBD">
      <fvRsCtx tnFvCtxName="mgmt_ctx1"/>
      <fvSubnet ip="3.3.3.0/24"/>
    </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="tenant1">
    <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-tenant1/ap-services/epg-ifc"/>
    </vnsLDevVip>
    <vnsCMgmt name="devMgmt"
      host="3.3.3.180"
      port="80"/>
    <vnsCCred name="username"
      value="nsroot"/>
    <vnsCCredSecret name="password"
      value="nsroot"/>
  </fvTenant>
</polUni>
```

Configuring In-Band Connectivity to Devices Using Management Tenant VRF Using the REST API

The following is an example of using REST APIs to configure in-band connectivity to devices 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"/>
    </vnsLDevVip>
    <vnsCMgmt name="devMgmt">
    </vnsCMgmt>
  </fvTenant>
</polUni>
```

```

        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.

Device Packages

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 single class of service devices and provides the APIC with information about the device and its capabilities.

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

Notes for Installing a Device Package with the REST APIs

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

Uploading a Device Package File Using the API

To install a service device, you must upload a device package file to APIC. The API command for this operation uses a special form of URI:

```
{ http | https } :// host [:port] /ppi /node /mo / . { json | xml }
```

The URI path contains 'ppi' (package programming interface) instead of 'api', and the command is sent as a POST operation with the device package file as the body of the message. The device package file is a zip file.

This example shows an API operation that uploads a device package file:

```
POST https://192.0.20.123/ppi/node/mo/.json
```

For more information about installing L4-L7 service device packages, see *Cisco APIC Layer 4 to Layer 7 Services Deployment Guide*.

Installing a Device Package Using the REST API

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

Install the device package.

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

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

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

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

The message must have a valid session cookie.

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

Using an Imported Device with the REST APIs

The following REST API uses an imported device:

```
<polUni>
  <fvTenant dn="uni/tn-tenant1" name="tenant1">
    <vnsLDevIf ldev="uni/tn-mgmt/lDevVip-ADCCluster1"/>
    <vnsLDevCtx ctrctNameOrLbl="any" graphNameOrLbl="any" nodeNameOrLbl="any">
      <vnsRsLDevCtxToLDev tDn="uni/tn-tenant1/lDevIf-[uni/tn-mgmt/lDevVip-ADCCluster1]"/>
      <vnsLIfCtx connNameOrLbl="inside">
        <vnsRsLIfCtxToLIf
tDn="uni/tn-tenant1/lDevIf-[uni/tn-mgmt/lDevVip-ADCCluster1]/lDevIfLIf-inside"/>
          <fvSubnet ip="10.10.10.10/24"/>
          <vnsRsLIfCtxToBD tDn="uni/tn-tenant1/BD-tenant1BD1"/>
        </vnsLIfCtx>
        <vnsLIfCtx connNameOrLbl="outside">
          <vnsRsLIfCtxToLIf
tDn="uni/tn-tenant1/lDevIf-[uni/tn-mgmt/lDevVip-ADCCluster1]/lDevIfLIf-outside"/>
          <fvSubnet ip="70.70.70.70/24"/>
        </vnsLIfCtx>
      </vnsLDevCtx>
    </vnsLDevIf>
  </fvTenant>
</polUni>
```

```

        <vnsRsLifCtxToBD tDn="uni/tn-tenant1/BD-tenant1BD4"/>
      </vnsLifCtx>
    </vnsLDevCtx>
  </fvTenant>
</polUni>

```

Trunking

About Trunking

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

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

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

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

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

Before you begin

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

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

```

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

```

Device Selection Policies

About Device Selection Policies

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

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

Creating a Device Selection Policy Using the REST API

The following REST API creates a device selection policy:

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

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

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

Adding a Logical Interface in a Device Using the REST APIs

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

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

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

    <vnsLIf name="ext">

      <vnsRsMetaIf tDn="uni/infra/mDev-Acme-ADC-1.0/mIfLbl-outside"/>
      <vnsRsCIIfAtt 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/1DevVip-ADCCluster1/cDev-ADC1/cIf-int"/>
</vnsLIf>
</vnsLDevVip>
</fvTenant>
</polUni>
```

Policy Based Redirect and Service Nodes Tracking

Policy-Based Redirect and Tracking Service Nodes

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

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

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

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

You can use the following protocols for tracking:

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

Policy-Based Redirect and Threshold Settings for Tracking Service Nodes

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

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

- **Maximum threshold:** The maximum threshold percentage specified. Once the minimum threshold is reached, to get back to operational state, the maximum percentage must first be reached. The default value is 0.

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

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

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

- Beginning in release 4.0(1), remote leaf switch configurations support PBR tracking, but only if system-level global GIPo is enabled. See *Configuring Global GIPo for Remote Leaf Using the GUI*.
- Beginning in release 4.0(1), remote leaf switch configurations support PBR resilient hashing.
- A Cisco ACI Multi-Pod fabric setup is supported.
- A Cisco ACI Multi-Site setup is not supported.
- An L3Out is supported for the consumer and provider EPGs.
- TCP or ICMP protocol types are used to track the redirect destination nodes.
- PBR supports up to 100 trackable IP addresses in leaf switches and 200 trackable IP addresses in the Cisco Application Centric Infrastructure (ACI) fabric.
- PBR supports up to 1,000 service graph instances per Cisco ACI fabric.
- PBR supports up to 100 service graph instances per device.
- You can configure up to 40 service nodes per PBR policy.
- You can configure up to 3 service nodes per service chain.
- Shared services are supported with PBR tracking.
- The following threshold down actions are supported:
 - deny action
 - permit action
- If multiple PBR policies have the same PBR destination IP address in the same VRF instance, the policies must use the same IP SLA policy and health group for the PBR destination.

Configuring PBR to Support Tracking Service Nodes Using the REST API

Configure PBR to support tracking service nodes.

Example:

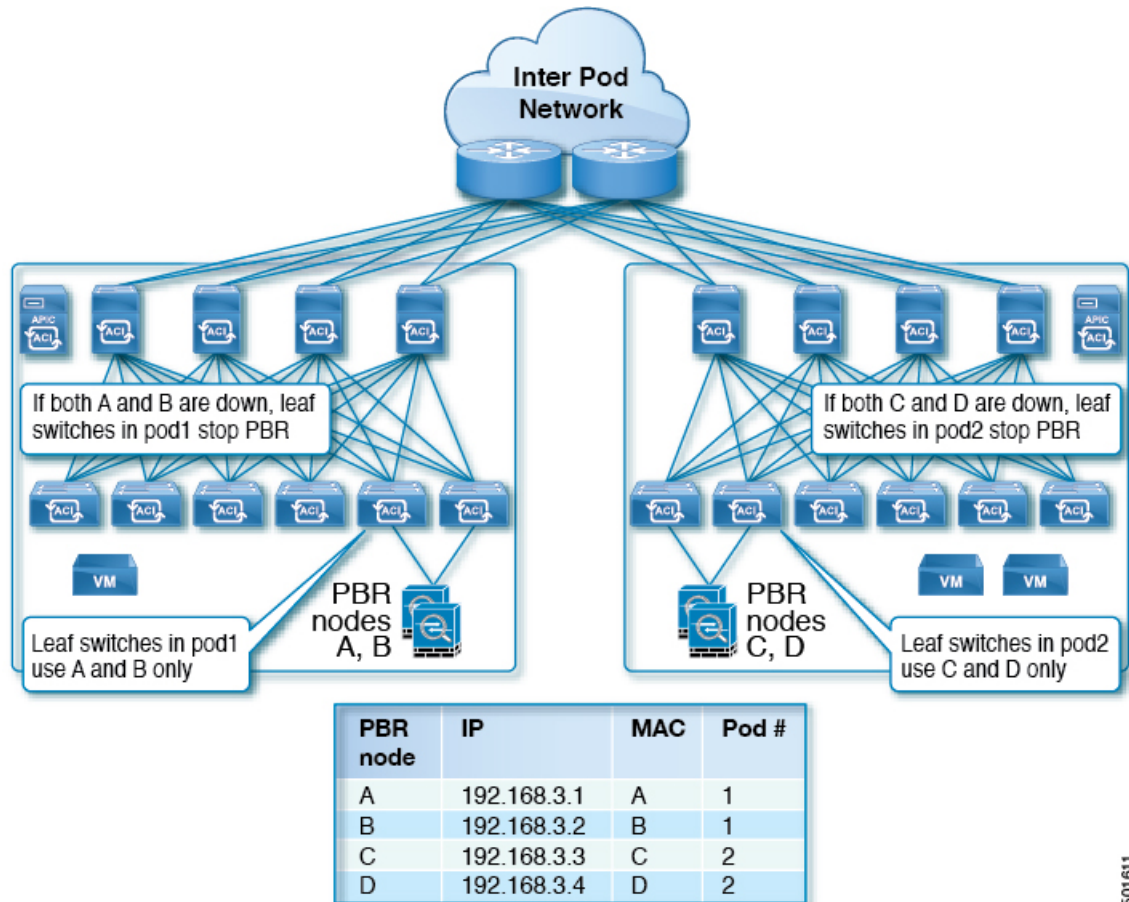
```
<polUni>
  <fvTenant name="t1" >
    <fvIPSLAMonitoringPol name="tcp_Freq60_Poll" slaType="tcp" slaFrequency="60" slaPort="2222" />
    <vnsSvcCont>
      <vnsRedirectHealthGroup name="fwService1"/>
      <vnsSvcRedirectPol name="fwExt" hashingAlgorithm="sip" thresholdEnable="yes"
        minThresholdPercent="20" maxThresholdPercent="80">
        <vnsRedirectDest ip="40.40.40.100" mac="00:00:00:00:00:01">
          <vnsRsRedirectHealthGroup tDn="uni/tn-t1/svcCont/redirectHealthGroup-fwService1"/>
        </vnsRedirectDest>
        <vnsRsIPSLAMonitoringPol tDn="uni/tn-t1/ipslaMonitoringPol-tcp_Freq60_Poll"/>
      </vnsSvcRedirectPol>
      <vnsSvcRedirectPol name="fwInt" hashingAlgorithm="sip" thresholdEnable="yes"
        minThresholdPercent="20" maxThresholdPercent="80">
        <vnsRedirectDest ip="30.30.30.100" mac="00:00:00:00:00:02">
          <vnsRsRedirectHealthGroup tDn="uni/tn-t1/svcCont/redirectHealthGroup-fwService1"/>
        </vnsRedirectDest>
        <vnsRsIPSLAMonitoringPol tDn="uni/tn-t1/ipslaMonitoringPol-tcp_Freq60_Poll"/>
      </vnsSvcRedirectPol>
    </vnsSvcCont>
  </fvTenant>
</polUni>
```

About Location-Aware Policy Based Redirect

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

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

Figure 51: An Example of Location Aware PBR Configuration with Two Pods



501611

Guidelines for Location-Aware PBR

Follow these guidelines when using location-aware PBR:

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

Use location-aware PBR for a contract that is enforced on the same leaf nodes for incoming and returning traffic, such as an intra-VRF contract for external-EPG-to-EPG and an inter-VRF contract for EPG-to-EPG traffic. Otherwise, there can be a loss of traffic symmetry.

- If multiple PBR policies have the same PBR destination IP address in the same VRF, then all of the policies must either have Pod ID aware redirection enabled or Pod ID aware redirection disabled. The same (VRF, IP address) pair cannot be used in Pod ID aware redirection enabled and Pod ID aware redirection disabled policies at the same time. For example, the following configuration is not supported:
 - PBR-policy1 has PBR destination 192.168.1.1 in VRF A, Pod ID aware redirection enabled, and 192.168.1.1 is set to POD 1.
 - PBR-policy2 has PBR destination 192.168.1.1 in VRF A and Pod ID aware redirection disabled.

Configuring Location-Aware PBR Using the REST API

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

Configure location-aware PBR.

Example:

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

About Layer 1/Layer 2 Policy-Based Redirect

Using a Layer 1 device is typically referred to as *inline mode* or *wire mode* and is used for firewalls and intrusion prevention systems (IPS) if the service device is expected to perform security functions that are not participating in Layer 2 or Layer 3 forwarding.

Using a Layer 2 device is typically referred to as *transparent mode* or *bridged mode* and is used for firewalls and IPS.

Using a Layer 3 device is typically referred to as *routed mode* and is used for router firewalls and load balancers.

Prior to Cisco Application Policy Infrastructure Controller(APIC) release 4.1, PBR could be configured to redirect traffic to a Layer 4 to Layer 7 services device configured only in Layer 3 device (Go-To) mode. If the Layer 4 to Layer 7 services device is a Layer 1 or Layer 2 device, such as a transparent firewall, PBR could not be used. You could only deploy a Layer 4 to Layer 7 services device operating in Layer 1 or Layer 2 mode by using a service graph and defining the Layer 4 to Layer 7 services device in **Go-Through** mode.

Beginning with Cisco APIC release 4.1, PBR can be configured to redirect traffic to a Layer 4 to Layer 7 services device configured in the Layer 1/Layer 2 device mode as well. PBR can be used with inline IPS or a transparent firewall, in addition to a routed mode firewall.

As part of the Layer 1/Layer 2 PBR feature, the Cisco APIC can verify whether the Layer 4 to Layer 7 services device is forwarding traffic by using Layer 2 ping packets for link layer tracking.

Unlike the **Go-Through** mode, which can also forward non-IP address traffic, Layer 1/Layer 2 PBR is applicable only to IP address traffic.

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

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

- Layer 1/Layer 2 PBR is not supported from the CLI.
- Active-active deployment/ECMP paths are not supported for Layer 1/Layer 2 PBR devices.
- The two legs of the Layer 1 service device need to be configured on a different leaf switch to avoid packet loops. Per port VLAN is not supported.
- A shared bridge domain is not supported. A Layer 1/Layer 2 device bridge domain cannot be shared with Layer 3 device or regular EPGs.
- A service node in managed mode is not supported.
- Layer 1/Layer 2 devices support physical domains only. VMM domains are not supported.
- As active-active is not supported, the threshold is not applicable. The down action is `deny` when tracking is enabled. The down action `permit` can not be set.
- Tracking is mandatory when service devices are in active/standby HA mode.
- For both Layer 1 and Layer 2 PBR, the MAC address is an optional parameter. Cisco APIC assigns the MAC address if it is not configured. In both cases, traffic is routed to the device using the MAC address, which can be user configured or implicitly assigned by Cisco APIC.
- Beginning with the Cisco APIC release 4.1, Layer 1/Layer 2 PBR supports redirection to transparent services.

Configuring Layer 1/ Layer 2 PBR Using the REST API

Layer 1/ Layer 2 Policy-Based Redirect configuration:

Example:

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

    <!--If L1/L2 device in active-active mode -- >
    <vnsLDevVip name="N1" activeActive="yes" funcType="L1" managed="no">
    </vnsLDevVip>
    <!--If L1/L2 device in active-standby mode -- >
    <vnsLDevVip name="N1" activeActive="no" funcType="L1" managed="no">
    </vnsLDevVip>

    <vnsAbsGraph descr="" dn="uni/tn-coke/AbsGraph-WebGraph" name="WebGraph" ownerKey="" ownerTag=""
    uiTemplateType="UNSPECIFIED">

    <!--For L2 device -- >
```

```

    <vnsAbsNode descr="" funcTemplateType="OTHER" funcType="L2" isCopy="no" managed="no" name="N1"
ownerKey="" ownerTag="" routingMode="Redirect" sequenceNumber="0" shareEncap="no">
    </vnsAbsNode>

    <!--For L1 device -->
    <vnsAbsNode descr="" funcTemplateType="OTHER" funcType="L1" isCopy="no" managed="no" name="N1"
ownerKey="" ownerTag="" routingMode="Redirect" sequenceNumber="0" shareEncap="no">
    </vnsAbsNode>

</vnsAbsGraph>

<fvIPSLAMonitoringPol name="Pol2" slaType="l2ping"/>
<vnsSvcCont>
<vnsRedirectHealthGroup name="2" />
    <vnsSvcRedirectPol name="N1Ext" destType="L2">
        <vnsRsIPSLAMonitoringPol tDn="uni/tn-coke/ipslaMonitoringPol-Pol2"/>
    <vnsL1L2RedirectDest destName="1">
        <vnsRsL1L2RedirectHealthGroup tDn="uni/tn-coke/svcCont/redirectHealthGroup-2"/>
    <vnsRsToCif tDn="uni/tn-coke/lDevVip-N1/cDev-ASA1/cIf-[Gig0/0]"/>
</vnsL1L2RedirectDest>
    </vnsSvcRedirectPol>

    <vnsSvcRedirectPol name="N1Int" destType="L2">
        <vnsRsIPSLAMonitoringPol tDn="uni/tn-coke/ipslaMonitoringPol-Pol2"/>
    <vnsL1L2RedirectDest destName="2">
        <vnsRsL1L2RedirectHealthGroup tDn="uni/tn-coke/svcCont/redirectHealthGroup-2"/>
    <vnsRsToCif tDn="uni/tn-coke/lDevVip-N1/cDev-ASA1/cIf-[Gig0/1]"/>
</vnsL1L2RedirectDest>
    </vnsSvcRedirectPol>
</vnsSvcCont>
</fvTenant>
</polUni>

```

Service Graph Templates

About Service Graph Templates

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

Cisco ACI treats services as an integral part of an application. Any services that are required are treated as a service graph that is instantiated on the Cisco ACI fabric from the Cisco Application Policy Infrastructure Controller (APIC). Users define the service for the application, while service graph templates identify the set of network or service functions that are needed by the application. Once the graph is configured in the Cisco APIC, the Cisco APIC automatically configures the services according to the service function requirements that are specified in the service graph template. The Cisco APIC also automatically configures the network

according to the needs of the service function that is specified in the service graph template, which does not require any change in the service device.

Configuring a Service Graph Template Using the REST APIs

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

```
<polUni>
  <fvTenant name="acme">
    <vnsAbsGraph name="G1">
      <vnsAbsTermNodeCon name="Input1">
        <vnsAbsTermConn name="C1">
          </vnsAbsTermConn>
        </vnsAbsTermNodeCon>
      <vnsAbsNode name="Node" funcType="GoTo">
        <vnsRsDefaultScopeToTerm
          tDn="uni/tn-acme/AbsGraph-G1/AbsTermNodeProv-Output1/outtmn1"/>
        <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"/>
      </vnsAbsConnection>
      <vnsRsAbsConnectionConns
        tDn="uni/tn-acme/AbsGraph-G1/AbsTermNodeProv-Output1/AbsTConn"/>
    </vnsAbsGraph>
  </fvTenant>
</polUni>
```

Creating a Security Policy Using the REST APIs

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

```
<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>
```

Layer 4 to Layer 7 Parameters

About Modifying the Configuration Parameters of a Deployed Service Graph

When you first deploy a service graph, the configuration parameters or functions for the service graph must be defined before you can successfully deploy the service graph. These configuration parameters or functions include device network configurations, such as IP addresses, route prefix, and next hop information, as well as the services configuration, such as the IP access list for a firewall or server load balancing configuration for a load balancer.

You must modify the service graph function as part of the day-to-day operation of the Application Policy Infrastructure Controller (APIC). You can modify a service graph's configuration parameters and functions by using the GUI or CLI of the APIC. Modifying functions of a service device through the APIC does not require changes on a service device.

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" />
  </vnsFolderInst>
</fvAEPg>
```

```

    <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>
</vnsFolderInst>
<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="vsservername" key="vsservername" 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>

```

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>

```

Example XML POST for an Abstract Function Node With Configuration Parameters

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

```

<vnsAbsNode name = "SLB" funcType="GoTo" >
  <vnsRsDefaultScopeToTerm tDn="uni/tn-tenant1/AbsGraph-G3/AbsTermNode-Output1/outtmn1"/>

  <vnsAbsFuncConn name = "C4" direction = "input">
    <vnsRsMConnAtt tDn="uni/infra/mDev-Acme-ADC-1.0/mFunc-SLB/mConn-external" />
  </vnsAbsFuncConn>
  <vnsAbsFuncConn name = "C5" direction = "output">
    <vnsRsMConnAtt tDn="uni/infra/mDev-Acme-ADC-1.0/mFunc-SLB/mConn-internal" />
  </vnsAbsFuncConn>

  <vnsAbsDevCfg>
    <vnsAbsFolder key="Network" name="Network" scopedBy="epg">
      <!-- Following scopes this folder to input terminal or Src Epg -->
      <vnsRsScopeToTerm tDn="uni/tn-tenant1/AbsGraph-G3/AbsTermNode-Output1/outtmn1"/>

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

      <!-- SNIP address -->

```

```

    <vnsAbsFolder key="snip" name="snip" scopedBy="epg">
      <vnsAbsParam name="snipaddress" key="snipaddress" value=""/>
    </vnsAbsFolder>

  </vnsAbsFolder>

  <vnsAbsFolder key="Service" name="Service" scopedBy="epg" cardinality="n">
    <vnsRsScopeToTerm tDn="uni/tn-tenant1/AbsGraph-G3/AbsTermNode-Output1/outtmnl"/>

    <vnsAbsParam name="servicename" key="servicename" value=""/>
    <vnsAbsParam name="servername" key="servername" value=""/>
    <vnsAbsParam name="serveripaddress" key="serveripaddress" value=""/>
  </vnsAbsFolder>
</vnsAbsDevCfg>

<vnsAbsFuncCfg>
  <vnsAbsFolder key="VServer" name="VServer" scopedBy="epg">
    <vnsRsScopeToTerm tDn="uni/tn-tenant1/AbsGraph-G3/AbsTermNode-Output1/outtmnl"/>

    <!-- Virtual Server Configuration -->
    <vnsAbsParam name="vip" key="vip" value=""/>
    <vnsAbsParam name="vservername" key="vservername" value=""/>
    <vnsAbsParam name="servicename" key="servicename" value=""/>
    <vnsRsCfgToConn tDn="uni/tn-tenant1/AbsGraph-G3/AbsNode-Node2/AbsFConn-C4" />
  </vnsAbsFolder>
</vnsAbsFuncCfg>
<vnsRsNodeToMFunc tDn="uni/infra/mDev-Acme-ADC-1.0/mFunc-SLB"/>
</vnsAbsNode>

```

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>

```

```

        </vnsAbsFolder>
      </vnsAbsFuncCfg>
    </vnsAbsFuncProf>
  </vnsAbsFuncProfGrp>
</vnsAbsFuncProfContr>

```

Copy Services

About Copy Services

Unlike Switched Port Analyzer (SPAN), which duplicates all traffic, the Cisco Application Centric Infrastructure (ACI) copy services feature enables selectively copying portions of the traffic between endpoint groups, according to the specifications of the contract. Broadcast, unknown unicast and multicast (BUM), and control plan traffic not covered by the contract are not copied. In contrast, SPAN copies everything out of endpoint groups, access ports, or uplink ports. Unlike SPAN, copy services do not add headers to the copied traffic. Copy service traffic is managed internally in the switch to minimize impact on normal traffic forwarding.

For more information about deploying Layer 4 to Layer 7 services, see the *Cisco APIC Layer 4 to Layer 7 Services Deployment Guide*.

Configuring Copy Services Using the REST API

A copy device is used as part of the copy services feature to create a copy node. A copy node specifies at which point of the data flow between endpoint groups to copy traffic.

This procedure provides examples of using the REST API to configure copy services.



Note When you configure a copy device, the context aware parameter is not used. The context aware parameter has a default value of `single context`, which can be ignored.

Before you begin

You must have configured a tenant.

Step 1 Create a copy device.

Example:

```

<vnsLDevVip contextAware="single-Context" devtype="PHYSICAL" funcType="None" isCopy="yes"
  managed="no" mode="legacy-Mode" name="copy0" packageModel="" svcType="COPY" trunking="no">
  <vnsRsALDevToPhysDomP tDn="uni/phys-phys_scale_copy"/>
  <vnsCDev devCtxLbl="" name="copy_Dyn_Device_0" vcenterName="" vmName="">
    <vnsCIf name="int1" vnicName="">
      <vnsRsCIfPathAtt tDn="topology/pod-1/paths-104/pathep-[eth1/15]"/>
    </vnsCIf>
    <vnsCIf name="int2" vnicName="">
      <vnsRsCIfPathAtt tDn="topology/pod-1/paths-105/pathep-[eth1/15]"/>
    </vnsCIf>
  </vnsCDev>
  <vnsLIf encap="vlan-3540" name="TAP">
    <vnsRsCIfAttN tDn="uni/tn-t22/1DevVip-copy0/cDev-copy_Dyn_Device_0/cIf-[int2]"/>
  </vnsLIf>
</vnsLDevVip>

```

```

    <vnsRsCIfAttN tDn="uni/tn-t22/lDevVip-copy0/cDev-copy_Dyn_Device_0/cIf-[int1]"/>
  </vnsLIf>
</vnsLDevVip>

```

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

Example:

```

<vnsLDevCtx ctrctNameOrLbl="c0" descr="" graphNameOrLbl="g0" name="" nodeNameOrLbl="CP1">
  <vnsRsLDevCtxToLDev tDn="uni/tn-t22/lDevVip-copy0"/>
  <vnsLIfCtx connNameOrLbl="copy" descr="" name="">
    <vnsRsLIfCtxToLIf tDn="uni/tn-t22/lDevVip-copy0/lIf-TAP"/>
  </vnsLIfCtx>
</vnsLDevCtx>

```

Step 3 Create and apply the copy graph template.

Example:

```

<vnsAbsGraph descr="" name="g0" ownerKey="" ownerTag="" uiTemplateType="UNSPECIFIED">
  <vnsAbsTermNodeCon descr="" name="T1" ownerKey="" ownerTag="">
    <vnsAbsTermConn attNotify="no" descr="" name="1" ownerKey="" ownerTag=""/>
    <vnsInTerm descr="" name=""/>
    <vnsOutTerm descr="" name=""/>
  </vnsAbsTermNodeCon>
  <vnsAbsTermNodeProv descr="" name="T2" ownerKey="" ownerTag="">
    <vnsAbsTermConn attNotify="no" descr="" name="1" ownerKey="" ownerTag=""/>
    <vnsInTerm descr="" name=""/>
    <vnsOutTerm descr="" name=""/>
  </vnsAbsTermNodeProv>
  <vnsAbsConnection adjType="L2" connDir="provider" connType="external" descr="" name="C1"
    ownerKey="" ownerTag="" unicastRoute="yes">
    <vnsRsAbsConnectionConns tDn="uni/tn-t22/AbsGraph-g0/AbsTermNodeCon-T1/AbsTConn"/>
    <vnsRsAbsConnectionConns tDn="uni/tn-t22/AbsGraph-g0/AbsTermNodeProv-T2/AbsTConn"/>
    <vnsRsAbsCopyConnection tDn="uni/tn-t22/AbsGraph-g0/AbsNode-CP1/AbsFConn-copy"/>
  </vnsAbsConnection>
  <vnsAbsNode descr="" funcTemplateType="OTHER" funcType="None" isCopy="yes" managed="no"
    name="CP1" ownerKey="" ownerTag="" routingMode="unspecified" sequenceNumber="0"
    shareEncap="no">
    <vnsAbsFuncConn attNotify="no" descr="" name="copy" ownerKey="" ownerTag=""/>
    <vnsRsNodeToLDev tDn="uni/tn-t22/lDevVip-copy0"/>
  </vnsAbsNode>
</vnsAbsGraph>

```

Step 4 Define the relation to the copy graph in the contract that is associated with the endpoint groups.

Example:

```

<vzBrCP descr="" name="c0" ownerKey="" ownerTag="" prio="unspecified" scope="tenant"
  targetDscp="unspecified">
  <vzSubj consMatchT="AtleastOne" descr="" name="Subject" prio="unspecified"
    provMatchT="AtleastOne" revFltPorts="yes" targetDscp="unspecified">
    <vzRsSubjFiltAtt directives="" tnVzFilterName="default"/>
    <vzRsSubjGraphAtt directives="" tnVnsAbsGraphName="g0"/>
  </vzSubj>
</vzBrCP>

```

Step 5 Attach the contract to the endpoint group.

Example:

```

<fvAEPg name="epg2860">
  <fvRsCons tnVzBrCPName="c0"/>
  <fvRsBd tnFvBDName="bd0"/>
  <fvRsDomAtt tDn="uni/phys-phys_scale_SB"/>
  <fvRsPathAtt tDn="topology/pod-1/paths-104/pathep-[PC_int2_g1]" encap="vlan-2860">

```

```
    instrImedcy="immediate"/>
</fvAEPg>
<fvAEPg name="epg2861">
  <fvRsProv tnVzBrCPName="c0"/>
  <fvRsBd tnFvBDName="bd0"/>
  <fvRsDomAtt tDn="uni/phys-phys_scale_SB"/>
  <fvRsPathAtt tDn="topology/pod-1/paths-105/pathep-[PC_policy]" encap="vlan-2861"
    instrImedcy="immediate"/>
</fvAEPg>
```

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 Cisco APIC GUI can also be done using XML-based REST POSTs using the northbound APIs. For example, you can monitor events through those APIs, dynamically enable EPGs, and add policies.

You can also use the northbound REST APIs to monitor for notifications that a device has been brought onboard, and to monitor faults. In both cases, you can monitor events that trigger specific actions. For example, if you see faults that occur on a specific application tier and determine that there is a loss of connectivity and a leaf node is going down, you can trigger an action to redeploy those applications somewhere else. If you have certain contracts on which you detect packet drops occurring, you could enable some copies of those contracts on the particular application. You can also use a statistics monitoring policy, where you monitor certain counters because of issues that have been reported.

For information on how to construct the XML files submitted to the Cisco APIC northbound API, see *Cisco APIC Layer 4 to Layer 7 Device Package Development Guide*.

The following Python APIs, defined in the *Cisco APIC Management Information Model Reference* can be used to submit REST POST calls using the northbound API:

- `vns:LDevVip`: Upload a device cluster
- `vns:CDev`: Upload a device
- `vns:LIf`: Create logical interfaces
- `vns:AbsGraph`: Create a graph
- `vz:BrCP`: Attach a graph to a contract

Examples of Automating Using the REST APIs

This section contains examples of using the REST APIs to automate tasks.

The following REST request creates a tenant with a broadcast domain, a Layer 3 network, application endpoint groups, and an application profile:

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

```

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

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

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

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

  <fvAEPg dn="uni/tn-acme/ap-MyAP/epg-MyClnt" name="MyClnt">
    <fvRsBd tnFvBDName="MySrvrBD"/>
    <fvRsDomAtt tDn="uni/vmmp-Vendor1/dom-MyVMs"/>
    <fvRsProv tnVzBrCPName="webCtrct"> </fvRsProv>
    <fvRsPathAtt tDn="topology/pod-1/paths-17/pathep-[eth1/21]"
      encap="vlan-202"/>
    <fvRsPathAtt tDn="topology/pod-1/paths-18/pathep-[eth1/21]"
      encap="vlan-202"/>
  </fvAEPg>

  <fvAEPg dn="uni/tn-acme/ap-MyAP/epg-MySRVR" name="MySRVR">
    <fvRsBd tnFvBDName="MyClntBD"/>
    <fvRsDomAtt tDn="uni/vmmp-Vendor1/dom-MyVMs"/>
    <fvRsCons tnVzBrCPName="webCtrct"> </fvRsCons>
    <fvRsPathAtt tDn="topology/pod-1/paths-17/pathep-[eth1/21]"
      encap="vlan-203"/>
    <fvRsPathAtt tDn="topology/pod-1/paths-18/pathep-[eth1/21]"
      encap="vlan-203"/>
  </fvAEPg>
</fvAp>
</fvTenant>
</polUni>

```

The following REST request creates a VLAN namespace:

```

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

```

The following REST request creates a VMM domain:

```

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

```

The following REST request creates a physical domain:

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

The following REST request creates a managed device cluster:

```
<polUni>
  <fvTenant dn="uni/tn-acme" name="acme">
    <vnsLDevVip name="ADCCluster1" contextAware=1>
      <vnsRsMDevAtt tDn="uni/infra/mDev-Acme-ADC-1.0"/>
      <vnsRsDevEpg tDn="uni/tn-acme/ap-services/epg-ifc"/>
      <vnsRsALDevToPhysDomP tDn="uni/phys-phys"/>
      <vnsCMgmt name="devMgmt" host="42.42.42.100" port="80"/>
      <vnsCCred name="username" value="admin"/>
      <vnsCCredSecret name="password" value="admin"/>
    </vnsLDevVip>
  </fvTenant>
</polUni>
```

The following REST request creates an unmanaged device cluster:

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

The following REST request creates a device cluster context:

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

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

```
<polUni>
  <fvTenant dn="uni/tn-coke{{tenantId}}" name="coke{{tenantId}}">
    <vnsRtrCfg name="Dev1Ctx1" rtrId="180.0.0.12"/>
    <vnsLDevCtx ctrctNameOrLbl="webCtrct1" graphNameOrLbl="WebGraph"
      nodeNameOrLbl="FW">
      <vnsRsLDevCtxToLDev tDn="uni/tn-tenant1/lDevVip-Firewall"/>
      <vnsRsLDevCtxToRtrCfg tnVnsRtrCfgName="FwRtrCfg"/>
      <vnsLIfCtx connNameOrLbl="internal">
        <vnsRsLIfCtxToInstP tDn="uni/tn-tenant1/out-OspfInternal/instP-IntInstP"
          status="created,modified"/>
        <vnsRsLIfCtxToLIf tDn="uni/tn-tenant1/lDevVip-Firewall/lIf-internal"/>
      </vnsLIfCtx>
      <vnsLIfCtx connNameOrLbl="external">
        <vnsRsLIfCtxToInstP tDn="uni/tn-common/out-OspfExternal/instP-ExtInstP">
```

```

        status="created,modified"/>
        <vnsRsLIfCtxToLIf tDn="uni/tn-tenant1/lDevVip-Firewall/lIf-external"/>
    </vnsLIfCtx>
</vnsLDevCtx>
</fvTenant>
</polUni>

```



Note For information about configuring external connectivity for tenants (a Layer 3 outside), see the *Cisco APIC Basic Configuration Guide*.

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

```

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

```

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

```

<polUni>
  <fvTenant dn="uni/tn-acme" name="acme">
    <vnsLDevVip name="ADCCluster1">
      <vnsCDev name="ADC1" devCtxLbl="C1">
        <vnsCIf name="int">
          <vnsRsCIfPathAtt tDn="topology/pod-1/paths-17/pathep-[eth1/22]"/>
        </vnsCIf>
        <vnsCIf name="ext">
          <vnsRsCIfPathAtt tDn="topology/pod-1/paths-17/pathep-[eth1/21]"/>
        </vnsCIf>
        <vnsCIf name="mgmt">
          <vnsRsCIfPathAtt tDn="topology/pod-1/paths-17/pathep-[eth1/20]"/>
        </vnsCIf>
        <vnsCMgmt name="devMgmt" host="172.30.30.100" port="80"/>
        <vnsCCred name="username" value="admin"/>
        <vnsCCredSecret name="password" value="admin"/>
      </vnsCDev>
      <vnsCDev name="ADC2" devCtxLbl="C2">
        <vnsCIf name="int">
          <vnsRsCIfPathAtt tDn="topology/pod-1/paths-17/pathep-[eth1/23]"/>
        </vnsCIf>
        <vnsCIf name="ext">
          <vnsRsCIfPathAtt tDn="topology/pod-1/paths-17/pathep-[eth1/24]"/>
        </vnsCIf>
        <vnsCIf name="mgmt">
          <vnsRsCIfPathAtt tDn="topology/pod-1/paths-17/pathep-[eth1/30]"/>
        </vnsCIf>
        <vnsCMgmt name="devMgmt" host="172.30.30.200" port="80"/>
        <vnsCCred name="username" value="admin"/>
        <vnsCCredSecret name="password" value="admin"/>
      </vnsCDev>
    </vnsLDevVip>

```



```

    </fvTenant>
  </polUni>

```

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

```

<polUni>
  <fvTenant dn="uni/tn-coke5" name="coke5">
    <vnsLDevVip name="Firewall15" devtype="VIRTUAL">
      <vnsCDev name="ASA5" vcenterName="vcenter1" vmName="ifav16-ASAv-scale-05">
        <vnsCIIf name="Gig0/0" vnicName="Network adapter 2"/>
        <vnsCIIf name="Gig0/1" vnicName="Network adapter 3"/>
        <vnsCIIf name="Gig0/2" vnicName="Network adapter 4"/>
        <vnsCIIf name="Gig0/3" vnicName="Network adapter 5"/>
        <vnsCIIf name="Gig0/4" vnicName="Network adapter 6"/>
        <vnsCIIf name="Gig0/5" vnicName="Network adapter 7"/>
        <vnsCIIf name="Gig0/6" vnicName="Network adapter 8"/>
        <vnsCIIf name="Gig0/7" vnicName="Network adapter 9"/>
        <vnsCMgmt name="devMgmt" host="3.5.3.170" port="443"/>
        <vnsCCred name="username" value="admin"/>
        <vnsCCredSecret name="password" value="insieme"/>
      </vnsCDev>
    </vnsLDevVip>
  </fvTenant>
</polUni>

```

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

```

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

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

        <!-- Node1 Provides SLB functionality -->
        <vnsAbsNode name = "Node1" funcType="GoTo" >
          <vnsRsDefaultScopeToTerm
            tDn="uni/tn-acme/AbsGraph-G1/AbsTermNode-Output1/outtmn1"/>

          <vnsAbsFuncConn name = "C4" direction = "input">
            <vnsRsMConnAtt tDn="uni/infra/mDev-Acme-ADC-1.0/mFunc-SLB/mConn-external"/>

            <vnsRsConnToLIf tDn="uni/tn-acme/lDevVip-ADCCluster1/lIf-C4"/>
          </vnsAbsFuncConn>

          <vnsAbsFuncConn name = "C5" direction = "output">
            <vnsRsMConnAtt tDn="uni/infra/mDev-Acme-ADC-1.0/mFunc-SLB/mConn-internal"/>

            <vnsRsConnToLIf tDn="uni/tn-acme/lDevVip-ADCCluster1/lIf-C5"/>
          </vnsAbsFuncConn>

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

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

        <vnsAbsConnection name = "CON1">
          <vnsRsAbsConnectionConns
            tDn="uni/tn-acme/AbsGraph-G1/AbsTermNode-Input1/AbsTConn"/>
          <vnsRsAbsConnectionConns

```

```

        tDn="uni/tn-acme/AbsGraph-G1/AbsNode-Node1/AbsFConn-C4"/>
    </vnsAbsConnection>

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

```

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

```

<polUni>
  <fvTenant name="HA_Tenant1">
    <vnsAbsGraph name="g1">

      <vnsAbsTermNodeProv name="Input1">
        <vnsAbsTermConn name="C1">
        </vnsAbsTermConn>
      </vnsAbsTermNodeProv>

      <!-- Node1 Provides LoadBalancing functionality -->
      <vnsAbsNode name="Node1" managed="no">
        <vnsRsDefaultScopeToTerm
            tDn="uni/tn-HA_Tenant1/AbsGraph-g1/AbsTermNodeProv-Input1/outtmnl"/>
        <vnsAbsFuncConn name="outside" attNotify="true">
        </vnsAbsFuncConn>
        <vnsAbsFuncConn name="inside" attNotify="true">
        </vnsAbsFuncConn>
      </vnsAbsNode>

      <vnsAbsTermNodeCon name="Output1">
        <vnsAbsTermConn name="C6">
        </vnsAbsTermConn>
      </vnsAbsTermNodeCon>

      <vnsAbsConnection name="CON2" adjType="L3" unicastRoute="yes">
        <vnsRsAbsConnectionConns
            tDn="uni/tn-HA_Tenant1/AbsGraph-g1/AbsTermNodeCon-Output1/AbsTConn"/>
        <vnsRsAbsConnectionConns
            tDn="uni/tn-HA_Tenant1/AbsGraph-g1/AbsNode-Node1/AbsFConn-outside"/>
      </vnsAbsConnection>

      <vnsAbsConnection name="CON1" adjType="L2" unicastRoute="no">
        <vnsRsAbsConnectionConns
            tDn="uni/tn-HA_Tenant1/AbsGraph-g1/AbsNode-Node1/AbsFConn-inside"/>
        <vnsRsAbsConnectionConns
            tDn="uni/tn-HA_Tenant1/AbsGraph-g1/AbsTermNodeProv-Input1/AbsTConn"/>
      </vnsAbsConnection>

    </vnsAbsGraph>
  </fvTenant>
</polUni>

```

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

```

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

```

```

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

```

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

```

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

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

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

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

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

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

        <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any" nodeNameOrLbl="any"
          key="Service" name="Service1">
          <vnsParamInst name="servicename" key="servicename"
            value="crpvgrtst02-8010"/>
          <vnsParamInst name="servicetype" key="servicetype" value="TCP"/>
          <vnsParamInst name="servername" key="servername"
            value="s192.168.100.100"/>
          <vnsParamInst name="serveripaddress" key="serveripaddress"
            value="192.168.100.100"/>
          <vnsParamInst name="serviceport" key="serviceport" value="8080"/>
          <vnsParamInst name="svrtimeout" key="svrtimeout" value="9000"/>
          <vnsParamInst name="clttimeout" key="clttimeout" value="9000"/>
          <vnsParamInst name="usip" key="usip" value="NO"/>
          <vnsParamInst name="useproxyport" key="useproxyport" value=""/>
          <vnsParamInst name="cip" key="cip" value="ENABLED"/>
          <vnsParamInst name="cka" key="cka" value="NO"/>
          <vnsParamInst name="sp" key="sp" value="OFF"/>
          <vnsParamInst name="cmp" key="cmp" value="NO"/>
          <vnsParamInst name="maxclient" key="maxclient" value="0"/>
          <vnsParamInst name="maxreq" key="maxreq" value="0"/>
          <vnsParamInst name="tcpb" key="tcpb" value="NO"/>
          <vnsCfgRelInst name="MonitorConfig" key="MonitorConfig"
            targetName="monitor1"/>
        </vnsFolderInst>

        <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any"

```

```

nodeNameOrLbl="any" key="Network" name="Network">
  <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any"
    nodeNameOrLbl="any" key="vip" name="vip">
    <vnsParamInst name="vipaddress1" key="vipaddress"
      value="10.10.10.100"/>
    </vnsFolderInst>
  <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any"
    nodeNameOrLbl="any" devCtxLbl="C1" key="snip" name="snip1">
    <vnsParamInst name="snipaddress" key="snipaddress"
      value="192.168.1.100"/>
    </vnsFolderInst>
  <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any"
    nodeNameOrLbl="any" devCtxLbl="C2" key="snip" name="snip2">
    <vnsParamInst name="snipaddress" key="snipaddress"
      value="192.168.1.101"/>
    </vnsFolderInst>
  <vnsFolderInst ctrctNameOrLbl="any" graphNameOrLbl="any"
    nodeNameOrLbl="any" devCtxLbl="C3" key="snip" name="snip3">
    <vnsParamInst name="snipaddress" key="snipaddress"
      value="192.168.1.102"/>
    </vnsFolderInst>
</vnsFolderInst>

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

```

Example: Configuring Layer 4 to Layer 7 Services (Firewall)

Example: Configuring Layer 4 to Layer 7 Services Using the REST API

This topic shows the steps for configuring Layer 4 to Layer 7 services (ASA Firewall) using the REST API.

Before you begin

- Create the tenant to use the Layer 4 to Layer 7 services, with a Layer 3 outside network and bridge domains.
- Create application profiles.
- Configure a physical or VMM domain.
- Import and register the device packages and configure parameters for them.

Step 1 Create a Layer 4 to Layer 7 **ASAv** device package model, using XML such as the following example:

Example:

```
<vnsLDevVip trunking="no" svcType="FW"
packageModel="ASAv" name="ASAv" mode="legacy-Mode"
managed="yes" isCopy="no" funcType="GoTo"
dn="uni/tn-Tenant-test/lDevVip-ASAv" devtype="VIRTUAL"
contextAware="single-Context">

<vnsCCred name="username" value="admin"/>
<vnsRsMDevAtt tDn="uni/infra/mDev-CISCO-ASA-1.2"/>
<vnsCCredSecret name="password"/>
<vnsCMgmt name="" port="443" host="172.31.184.249"/>
<vnsRsALDevToDomP tDn="uni/vmmp-VMware/dom-ACI_vDS"/>
<vnsCDev name="Device1" vmName="ASAv-L3" vcenterName="vcenter" devCtxLbl="">
<vnsCCred name="username" value="admin"/>
<vnsCCredSecret name="password"/>
<vnsCMgmt name="" port="443" host="172.31.184.249"/>
<vnsCIf name="GigabitEthernet0/1" vnicName="Network adapter 3"/>
<vnsCIf name="GigabitEthernet0/0" vnicName="Network adapter 2"/>
<vnsRsCDevToCtrlrP tDn="uni/vmmp-VMware/dom-ACI_vDS/ctrlr-vcenter"/>
</vnsCDev>

<vnsLIIf name="provider" encap="unknown">
<vnsRsMetaIf tDn="uni/infra/mDev-CISCO-ASA-1.2/mIfLbl-internal" isConAndProv="no"/>
<vnsRsCIfAttN tDn="uni/tn-Tenant-test/lDevVip-ASAv/cDev-Device1/cIf-[GigabitEthernet0/1]"/>
</vnsLIIf>

<vnsLIIf name="consumer" encap="unknown">
<vnsRsMetaIf tDn="uni/infra/mDev-CISCO-ASA-1.2/mIfLbl-external" isConAndProv="no"/>
<vnsRsCIfAttN tDn="uni/tn-Tenant-test/lDevVip-ASAv/cDev-Device1/cIf-[GigabitEthernet0/0]"/>
</vnsLIIf>
</vnsLDevVip>
```

Step 2 Configure a Layer 4 to Layer 7 **FW-Graph** using XML such as the following example:

Example:

Example: Configuring Layer 4 to Layer 7 Services Using the REST API

```

<vnsAbsGraph uiTemplateType="UNSPECIFIED" ownerTag="" ownerKey="" name="FW-Graph"
dn="uni/tn-Tenant-test/AbsGraph-FW-Graph" descr="">

<vnsAbsTermNodeCon ownerTag="" ownerKey="" name="T1" descr="">
<vnsAbsTermConn ownerTag="" ownerKey="" name="1" descr="" attNotify="no"/>
<vnsInTerm name="" descr=""/>
<vnsOutTerm name="" descr=""/>
</vnsAbsTermNodeCon>

<vnsAbsTermNodeProv ownerTag="" ownerKey="" name="T2" descr="">
<vnsAbsTermConn ownerTag="" ownerKey="" name="1" descr="" attNotify="no"/>
<vnsInTerm name="" descr=""/>
<vnsOutTerm name="" descr=""/>
</vnsAbsTermNodeProv>

<vnsAbsConnection ownerTag="" ownerKey="" name="C1" descr="" unicastRoute="yes"
directConnect="no" connType="external" connDir="provider" adjType="L2">
<vnsRsAbsConnectionConns tDn="uni/tn-Tenant-test/AbsGraph-FW-Graph/AbsNode-N1/AbsFConn-consumer"/>
<vnsRsAbsConnectionConns tDn="uni/tn-Tenant-test/AbsGraph-FW-Graph/AbsTermNodeCon-T1/AbsTConn"/>
</vnsAbsConnection>

<vnsAbsConnection ownerTag="" ownerKey="" name="C2" descr="" unicastRoute="yes"
directConnect="no" connType="external" connDir="provider" adjType="L2">
<vnsRsAbsConnectionConns tDn="uni/tn-Tenant-test/AbsGraph-FW-Graph/AbsNode-N1/AbsFConn-provider"/>
<vnsRsAbsConnectionConns tDn="uni/tn-Tenant-test/AbsGraph-FW-Graph/AbsTermNodeProv-T2/AbsTConn"/>
</vnsAbsConnection>

<vnsAbsNode ownerTag="" ownerKey="" name="N1" descr="" shareEncap="no" sequenceNumber="0"
routingMode="unspecified" managed="yes" isCopy="no" funcType="GoTo" funcTemplateType="FW_ROUTED">
<vnsAbsFuncConn ownerTag="" ownerKey="" name="consumer" descr="" attNotify="no">
<vnsRsMConnAtt tDn="uni/infra/mDev-CISCO-ASA-1.2/mFunc-Firewall/mConn-external"/>
</vnsAbsFuncConn>

<vnsAbsFuncConn ownerTag="" ownerKey="" name="provider" descr="" attNotify="no">
<vnsRsMConnAtt tDn="uni/infra/mDev-CISCO-ASA-1.2/mFunc-Firewall/mConn-internal"/>
</vnsAbsFuncConn>
<vnsRsNodeToAbsFuncProf
tDn="uni/infra/mDev-CISCO-ASA-1.2/absFuncProfContr/absFuncProfGrp-WebServiceProfileGroup/absFuncProf-WebPolicyForRoutedMode"/>
<vnsRsNodeToLDev tDn="uni/tn-Tenant-test/lDevVip-ASAv"/>
<vnsRsNodeToMFunc tDn="uni/infra/mDev-CISCO-ASA-1.2/mFunc-Firewall"/>
</vnsAbsNode>
</vnsAbsGraph>

```

Step 3 Create a device selection policy, using XML such as the following example:

Example:

```

<vnsLDevCtx nodeNameOrLbl="N1" name="" graphNameOrLbl="FW-Graph"
dn="uni/tn-Tenant-test/ldevCtx-c-Client-to-Web-g-FW-Graph-n-N1" descr="" ctrctNameOrLbl="Client-to-Web">
<vnsRsLDevCtxToLDev tDn="uni/tn-Tenant-test/lDevVip-ASAv"/>

<vnsLIfCtx name="" descr="" connNameOrLbl="provider">
<vnsRsLIfCtxToBD tDn="uni/tn-Tenant-test/BD-BD2"/>
<vnsRsLIfCtxToLIf tDn="uni/tn-Tenant-test/lDevVip-ASAv/lIf-provider"/>
</vnsLIfCtx>

<vnsLIfCtx name="" descr="" connNameOrLbl="consumer">
<vnsRsLIfCtxToBD tDn="uni/tn-Tenant-test/BD-BD1"/>
<vnsRsLIfCtxToLIf tDn="uni/tn-Tenant-test/lDevVip-ASAv/lIf-consumer"/>
</vnsLIfCtx>
</vnsLDevCtx>

```

Step 4 Configure a contract, associated with the **FW-Graph** service graph template, using XML such as the following example:

Example:

```
<vzBrCP targetDscp="unspecified" scope="tenant" prio="unspecified" ownerTag=""
ownerKey="" name="Client-to-Web" dn="uni/tn-Tenant-test/brc-Client-to-Web" descr="">

<vzSubj targetDscp="unspecified" prio="unspecified" name="Subject" descr=""
revFltPorts="yes" provMatchT="AtleastOne" consMatchT="AtleastOne"
<vzRsSubjFiltAtt tnVzFilterName="default" directives=""/>
<vzRsSubjGraphAtt directives="" tnVnsAbsGraphName="FW-Graph"/>
</vzSubj>
</vzBrCP>
```

Step 5 Create the **Client** EPG, using XML such as the following example:

Example:

```
<fvAEPg prio="unspecified" pcEnfPref="unenforced" name="Client"
matchT="AtleastOne" isAttrBasedEPg="no" fwdCtrl="" dn="uni/tn-Tenant-test/ap-ANP/epg-Client" descr="">
<fvRsCons prio="unspecified" tnVzBrCPName="Client-to-Web"/>
<fvRsDomAtt tDn="uni/vmmp-VMware/dom-ACI_vDS" resImedcy="lazy" primaryEncap="unknown"
instrImedcy="lazy" encap="unknown" delimiter="" classPref="encap"/>
<fvRsBd tnFvBDName="BD1"/>
<fvRsCustQosPol tnQosCustomPolName=""/>
</fvAEPg>
```

Step 6 Create the **Web** EPG, using XML such as the following example:

Example:

```
-<fvAEPg prio="unspecified" pcEnfPref="unenforced" name="Web" matchT="AtleastOne"
isAttrBasedEPg="no" fwdCtrl="" dn="uni/tn-Tenant-test/ap-ANP/epg-Web" descr="">
<fvRsDomAtt tDn="uni/vmmp-VMware/dom-ACI_vDS" resImedcy="lazy" primaryEncap="unknown"
instrImedcy="lazy" encap="unknown" delimiter="" classPref="encap"/>
<fvRsBd tnFvBDName="BD2"/>
<fvRsCustQosPol tnQosCustomPolName=""/>

<vnsFolderInst name="internalIf" scopedBy="epg" nodeNameOrLbl="N1" locked="no" key="Interface"
graphNameOrLbl="FW-Graph" devCtxLbl="" ctrctNameOrLbl="Client-to-Web" cardinality="unspecified">

<vnsFolderInst name="internalIfCfg" scopedBy="epg" nodeNameOrLbl="N1" locked="no" key="InterfaceConfig"
graphNameOrLbl="FW-Graph" devCtxLbl="" ctrctNameOrLbl="Client-to-Web" cardinality="unspecified">
<vnsParamInst name="internal_security_level" locked="no" key="security_level" cardinality="unspecified"
value="100" validation="" mandatory="no"/>
</vnsFolderInst>
</vnsFolderInst>

<vnsFolderInst name="externalIf" scopedBy="epg" nodeNameOrLbl="N1" locked="no" key="Interface"
graphNameOrLbl="FW-Graph" devCtxLbl="" ctrctNameOrLbl="Client-to-Web" cardinality="unspecified">

<vnsFolderInst name="ExtAccessGroup" scopedBy="epg" nodeNameOrLbl="N1" locked="no" key="AccessGroup"
graphNameOrLbl="FW-Graph" devCtxLbl="" ctrctNameOrLbl="Client-to-Web" cardinality="unspecified">
<vnsCfgRelInst name="name" locked="no" key="inbound_access_list_name" cardinality="unspecified"
mandatory="no"
targetName="access-list-inbound"/>
</vnsFolderInst>

<vnsFolderInst name="externalIfCfg" scopedBy="epg" nodeNameOrLbl="N1" locked="no" key="InterfaceConfig"
graphNameOrLbl="FW-Graph" devCtxLbl="" ctrctNameOrLbl="Client-to-Web" cardinality="unspecified">

<vnsParamInst name="external_security_level" locked="no" key="security_level"
cardinality="unspecified" value="50" validation="" mandatory="no"/>
</vnsFolderInst>
</vnsFolderInst>
```

Example: Configuring Layer 4 to Layer 7 Services Using the REST API

```

<vnsFolderInst name="IntConfig" scopedBy="epg" nodeNameOrLbl="N1" locked="no"
key="InIntfConfigRelFolder"
graphNameOrLbl="FW-Graph" devCtxLbl="" ctrctNameOrLbl="Client-to-Web" cardinality="unspecified">
<vnsCfgRelInst name="InConfigrel" locked="no" key="InIntfConfigRel"
cardinality="unspecified" mandatory="no" targetName="internalIf"/>
</vnsFolderInst>

<vnsFolderInst name="ExtConfig" scopedBy="epg" nodeNameOrLbl="N1" locked="no"
key="ExIntfConfigRelFolder"
graphNameOrLbl="FW-Graph" devCtxLbl="" ctrctNameOrLbl="Client-to-Web" cardinality="unspecified">
<vnsCfgRelInst name="ExtConfigrel" locked="no" key="ExIntfConfigRel" cardinality="unspecified"
mandatory="no"
targetName="externalIf"/>
</vnsFolderInst>

<vnsFolderInst name="access-list-inbound" scopedBy="epg" nodeNameOrLbl="N1" locked="no" key="AccessList"
graphNameOrLbl="FW-Graph" devCtxLbl="" ctrctNameOrLbl="Client-to-Web" cardinality="unspecified">
<vnsFolderInst name="permit-https" scopedBy="epg" nodeNameOrLbl="N1" locked="no"
key="AccessControlEntry"
graphNameOrLbl="FW-Graph" devCtxLbl="" ctrctNameOrLbl="Client-to-Web" cardinality="unspecified">
<vnsParamInst name="action-permit" locked="no" key="action" cardinality="unspecified" value="permit"
validation="" mandatory="no"/>
<vnsParamInst name="order1" locked="no" key="order" cardinality="unspecified" value="10" validation=""
mandatory="no"/>
</vnsFolderInst>

<vnsFolderInst name="dest-service" scopedBy="epg" nodeNameOrLbl="N1" locked="no"
key="destination_service"
graphNameOrLbl="FW-Graph" devCtxLbl="" ctrctNameOrLbl="Client-to-Web" cardinality="unspecified">
<vnsParamInst name="op" locked="no" key="operator" cardinality="unspecified" value="eq" validation=""
mandatory="no"/>
<vnsParamInst name="port" locked="no" key="low_port" cardinality="unspecified" value="https"
validation="" mandatory="no"/>
</vnsFolderInst>

<vnsFolderInst name="dest-address" scopedBy="epg" nodeNameOrLbl="N1" locked="no"
key="destination_address"
graphNameOrLbl="FW-Graph" devCtxLbl="" ctrctNameOrLbl="Client-to-Web" cardinality="unspecified">
<vnsParamInst name="any" locked="no" key="any" cardinality="unspecified" value="any" validation=""
mandatory="no"/>
</vnsFolderInst>

<vnsFolderInst name="src-address" scopedBy="epg" nodeNameOrLbl="N1" locked="no" key="source_address"
graphNameOrLbl="FW-Graph" devCtxLbl="" ctrctNameOrLbl="Client-to-Web" cardinality="unspecified">
<vnsParamInst name="any" locked="no" key="any" cardinality="unspecified" value="any" validation=""
mandatory="no"/>
</vnsFolderInst>

<vnsFolderInst name="tcp" scopedBy="epg" nodeNameOrLbl="N1" locked="no" key="protocol"
graphNameOrLbl="FW-Graph" devCtxLbl="" ctrctNameOrLbl="Client-to-Web" cardinality="unspecified">
<vnsParamInst name="tcp" locked="no" key="name_number" cardinality="unspecified" value="tcp"
validation="" mandatory="no"/>
</vnsFolderInst>
</vnsFolderInst>

<vnsFolderInst name="permit-http" scopedBy="epg" nodeNameOrLbl="N1" locked="no" key="AccessControlEntry"
graphNameOrLbl="FW-Graph" devCtxLbl="" ctrctNameOrLbl="Client-to-Web" cardinality="unspecified">
<vnsParamInst name="action-permit" locked="no" key="action" cardinality="unspecified" value="permit"
validation="" mandatory="no"/>

```



```

<vnsParamInst name="order1" locked="no" key="order" cardinality="unspecified" value="10" validation=""
mandatory="no"/>

<vnsFolderInst name="dest-service" scopedBy="epg" nodeNameOrLbl="N1" locked="no"
key="destination_service"
graphNameOrLbl="FW-Graph" devCtxLbl="" ctrctNameOrLbl="Client-to-Web" cardinality="unspecified">
<vnsParamInst name="op" locked="no" key="operator" cardinality="unspecified" value="eq" validation=""
mandatory="no"/>
<vnsParamInst name="port" locked="no" key="low_port" cardinality="unspecified" value="http"
validation="" mandatory="no"/>
</vnsFolderInst>

<vnsFolderInst name="dest-address" scopedBy="epg" nodeNameOrLbl="N1" locked="no"
key="destination_address"
graphNameOrLbl="FW-Graph" devCtxLbl="" ctrctNameOrLbl="Client-to-Web" cardinality="unspecified">
<vnsParamInst name="any" locked="no" key="any" cardinality="unspecified" value="any" validation=""
mandatory="no"/>
</vnsFolderInst>

<vnsFolderInst name="src-address" scopedBy="epg" nodeNameOrLbl="N1" locked="no" key="source_address"

graphNameOrLbl="FW-Graph" devCtxLbl="" ctrctNameOrLbl="Client-to-Web" cardinality="unspecified">
<vnsParamInst name="any" locked="no" key="any" cardinality="unspecified" value="any" validation=""
mandatory="no"/>
</vnsFolderInst>

<vnsFolderInst name="tcp" scopedBy="epg" nodeNameOrLbl="N1" locked="no" key="protocol"
graphNameOrLbl="FW-Graph" devCtxLbl="" ctrctNameOrLbl="Client-to-Web" cardinality="unspecified">

<vnsParamInst name="tcp" locked="no" key="name_number" cardinality="unspecified"
value="tcp" validation="" mandatory="no"/>
</vnsFolderInst>
</vnsFolderInst>
</vnsFolderInst>

<fvRsProv prio="unspecified" matchT="AtleastOne" tnVzBrCPName="Client-to-Web"/>
</fvAEPg>

```

Example

To configure the entire Layer 4 to Layer 7 ASAv firewall services for a tenant, use XML such as the following example;

```

<fvTenant ownerTag="" ownerKey="" name="Tenant-test" dn="uni/tn-Tenant-test" descr="">

<vnsLDevCtx name="" descr="" nodeNameOrLbl="N1" graphNameOrLbl="FW-Graph"
ctrctNameOrLbl="Client-to-Web"><vnsRsLDevCtxToLDev tDn="uni/tn-Tenant-test/lDevVip-ASAv"/>

<vnsLIfCtx name="" descr="" connNameOrLbl="provider">
<vnsRsLIfCtxToBD tDn="uni/tn-Tenant-test/BD-BD2"/>
<vnsRsLIfCtxToLIf tDn="uni/tn-Tenant-test/lDevVip-ASAv/lIf-provider"/>
</vnsLIfCtx>

<vnsLIfCtx name="" descr="" connNameOrLbl="consumer">
<vnsRsLIfCtxToBD tDn="uni/tn-Tenant-test/BD-BD1"/>
<vnsRsLIfCtxToLIf tDn="uni/tn-Tenant-test/lDevVip-ASAv/lIf-consumer"/>
</vnsLIfCtx>
</vnsLDevCtx>

```

Example: Configuring Layer 4 to Layer 7 Services Using the REST API

```

<vzBrCP ownerTag="" ownerKey="" name="Client-to-Web" descr="" targetDscp="unspecified"
scope="tenant" prio="unspecified">

<vzSubj name="Subject" descr="" targetDscp="unspecified" prio="unspecified" revFltPorts="yes"

provMatchT="AtleastOne" consMatchT="AtleastOne">
<vzRsSubjFiltAtt tnVzFilterName="default" directives=""/>
<vzRsSubjGraphAtt directives="" tnVnsAbsGraphName="FW-Graph"/>
</vzSubj>
</vzBrCP>

<vnsAbsGraph ownerTag="" ownerKey="" name="FW-Graph" descr="" uiTemplateType="UNSPECIFIED">
<vnsAbsTermNodeCon ownerTag="" ownerKey="" name="T1" descr="">
<vnsAbsTermConn ownerTag="" ownerKey="" name="1" descr="" attNotify="no"/>
<vnsInTerm name="" descr=""/>
<vnsOutTerm name="" descr=""/>
</vnsAbsTermNodeCon>

<vnsAbsTermNodeProv ownerTag="" ownerKey="" name="T2" descr="">
<vnsAbsTermConn ownerTag="" ownerKey="" name="1" descr="" attNotify="no"/>
<vnsInTerm name="" descr=""/>
<vnsOutTerm name="" descr=""/>
</vnsAbsTermNodeProv>

<vnsAbsConnection ownerTag="" ownerKey="" name="C1" descr="" unicastRoute="yes"
directConnect="no" connType="external" connDir="provider" adjType="L2">
<vnsRsAbsConnectionConns
tDn="uni/tn-Tenant-test/AbsGraph-FW-Graph/AbsNode-N1/AbsFConn-consumer"/>
<vnsRsAbsConnectionConns
tDn="uni/tn-Tenant-test/AbsGraph-FW-Graph/AbsTermNodeCon-T1/AbsTConn"/>
</vnsAbsConnection>

<vnsAbsConnection ownerTag="" ownerKey="" name="C2" descr="" unicastRoute="yes"
directConnect="no" connType="external" connDir="provider" adjType="L2">
<vnsRsAbsConnectionConns
tDn="uni/tn-Tenant-test/AbsGraph-FW-Graph/AbsNode-N1/AbsFConn-provider"/>
<vnsRsAbsConnectionConns
tDn="uni/tn-Tenant-test/AbsGraph-FW-Graph/AbsTermNodeProv-T2/AbsTConn"/>
</vnsAbsConnection>

<vnsAbsNode ownerTag="" ownerKey="" name="N1" descr="" shareEncap="no" sequenceNumber="0"
routingMode="unspecified" managed="yes" isCopy="no" funcType="GoTo"
funcTemplateType="FW_ROUTED">

<vnsAbsFuncConn ownerTag="" ownerKey="" name="consumer" descr="" attNotify="no">
<vnsRsMConnAtt tDn="uni/infra/mDev-CISCO-ASA-1.2/mFunc-Firewall/mConn-external"/>
</vnsAbsFuncConn>

<vnsAbsFuncConn ownerTag="" ownerKey="" name="provider" descr="" attNotify="no">
<vnsRsMConnAtt tDn="uni/infra/mDev-CISCO-ASA-1.2/mFunc-Firewall/mConn-internal"/>
</vnsAbsFuncConn>
<vnsRsNodeToAbsFuncProf
tDn="uni/infra/mDev-CISCO-ASA-1.2/absFuncProfContr/absFuncProfGrp-WebServiceProfileGroup/absFuncProf-WebPolicyForRoutedMode"/>
<vnsRsNodeToLDev tDn="uni/tn-Tenant-test/lDevVip-ASAv"/>
<vnsRsNodeToMFunc tDn="uni/infra/mDev-CISCO-ASA-1.2/mFunc-Firewall"/>
</vnsAbsNode>
</vnsAbsGraph>

<fvBD ownerTag="" ownerKey="" name="BD1" descr="" unicastRoute="yes" vmac="not-applicable"

unkMcastAct="flood" unkMacUcastAct="proxy" type="regular" multiDstPktAct="bd-flood"
mcastAllow="no"
mac="00:22:BD:F8:19:FF" llAddr="::" limitIpLearnToSubnets="no" ipLearning="yes"
epMoveDetectMode="" arpFlood="no">

```

```

<fvRsBDToNdp tnNdIfPolName=""/>
<fvRsCtx tnFvCtxName="VRF1"/>
<fvRsIgmprsn tnIgmprsnPolName=""/>
<fvRsBdToEpRet tnFvEpRetPolName="" resolveAct="resolve"/>
</fvBD>

<fvBD ownerTag="" ownerKey="" name="BD2" descr="" unicastRoute="yes" vmac="not-applicable"
unkMcastAct="flood" unkMacUcastAct="proxy" type="regular" multiDstPktAct="bd-flood"
mcastAllow="no"
mac="00:22:BD:F8:19:FF" llAddr=":" limitIpLearnToSubnets="no" ipLearning="yes"
epMoveDetectMode="" arpFlood="no">
<fvRsBDToNdp tnNdIfPolName=""/>
<fvRsCtx tnFvCtxName="VRF1"/>
<fvRsIgmprsn tnIgmprsnPolName=""/>
<fvRsBdToEpRet tnFvEpRetPolName="" resolveAct="resolve"/>
</fvBD>

<fvCtx ownerTag="" ownerKey="" name="VRF1" descr="" pcEnfPref="enforced" pcEnfDir="ingress"
knwMcastAct="permit">
<fvRsBgpCtxPol tnBgpCtxPolName=""/>
<fvRsCtxToExtRouteTagPol tnL3extRouteTagPolName=""/>
<fvRsOspfCtxPol tnOspfCtxPolName=""/>
<vzAny name="" descr="" matchT="AtleastOne"/>
<fvRsCtxToEpRet tnFvEpRetPolName=""/>
</fvCtx>
<vnsSvcCont/>

<fvAp ownerTag="" ownerKey="" name="ANP" descr="" prio="unspecified">

<fvAEPg name="Web" descr="" prio="unspecified" pcEnfPref="unenforced"
matchT="AtleastOne" isAttrBasedEPg="no" fwdCtrl="">
<fvRsDomAtt tDn="uni/vmmp-VMware/dom-ACI_vDS" resImedcy="lazy" primaryEncap="unknown"
instrImedcy="lazy" encap="unknown" delimiter="" classPref="encap"/>
<fvRsBd tnFvBDName="BD2"/>
<fvRsCustQosPol tnQosCustomPolName=""/>

<vnsFolderInst name="internalIf" nodeNameOrLbl="N1" graphNameOrLbl="FW-Graph"
ctrctNameOrLbl="Client-to-Web" scopedBy="epg" locked="no" key="interface"
devCtxLbl="" cardinality="unspecified">

<vnsFolderInst name="internalIfCfg" nodeNameOrLbl="N1" graphNameOrLbl="FW-Graph"
ctrctNameOrLbl="Client-to-Web" scopedBy="epg" locked="no" key="InterfaceConfig"
devCtxLbl="" cardinality="unspecified">
<vnsParamInst name="internal_security_level" locked="no" key="security_level"
cardinality="unspecified"
value="100" validation="" mandatory="no"/>
</vnsFolderInst>
</vnsFolderInst>

<vnsFolderInst name="externalIf" nodeNameOrLbl="N1" graphNameOrLbl="FW-Graph"
ctrctNameOrLbl="Client-to-Web" scopedBy="epg" locked="no" key="Interface"
devCtxLbl="" cardinality="unspecified">

<vnsFolderInst name="ExtAccessGroup" nodeNameOrLbl="N1" graphNameOrLbl="FW-Graph"
ctrctNameOrLbl="Client-to-Web" scopedBy="epg" locked="no" key="AccessGroup" devCtxLbl=""
cardinality="unspecified">
<vnsCfgRelInst name="name" locked="no" key="inbound_access_list_name"
cardinality="unspecified"
mandatory="no" targetName="access-list-inbound"/>
</vnsFolderInst>

<vnsFolderInst name="externalIfCfg" nodeNameOrLbl="N1" graphNameOrLbl="FW-Graph"

```

Example: Configuring Layer 4 to Layer 7 Services Using the REST API

```

ctrctNameOrLbl="Client-to-Web" scopedBy="epg" locked="no" key="InterfaceConfig"
devCtxLbl="" cardinality="unspecified">
<vnsParamInst name="external_security_level" locked="no" key="security_level"
cardinality="unspecified" value="50" validation="" mandatory="no"/>
</vnsFolderInst>
</vnsFolderInst>

<vnsFolderInst name="IntConfig" nodeNameOrLbl="N1" graphNameOrLbl="FW-Graph"
ctrctNameOrLbl="Client-to-Web" scopedBy="epg" locked="no" key="InIntfConfigRelFolder"
devCtxLbl="" cardinality="unspecified">
<vnsCfgRelInst name="InConfigrel" locked="no" key="InIntfConfigRel" cardinality="unspecified"
mandatory="no" targetName="internalIf"/>
</vnsFolderInst>

<vnsFolderInst name="ExtConfig" nodeNameOrLbl="N1" graphNameOrLbl="FW-Graph"
ctrctNameOrLbl="Client-to-Web" scopedBy="epg" locked="no" key="ExIntfConfigRelFolder"
devCtxLbl="" cardinality="unspecified">
<vnsCfgRelInst name="ExtConfigrel" locked="no" key="ExIntfConfigRel" cardinality="unspecified"
mandatory="no" targetName="externalIf"/>
</vnsFolderInst>

<vnsFolderInst name="access-list-inbound" nodeNameOrLbl="N1" graphNameOrLbl="FW-Graph"
ctrctNameOrLbl="Client-to-Web" scopedBy="epg" locked="no" key="AccessList" devCtxLbl=""
cardinality="unspecified">

<vnsFolderInst name="permit-https" nodeNameOrLbl="N1" graphNameOrLbl="FW-Graph"
ctrctNameOrLbl="Client-to-Web" scopedBy="epg" locked="no" key="AccessControlEntry"
devCtxLbl="" cardinality="unspecified">
<vnsParamInst name="action-permit" locked="no" key="action" cardinality="unspecified"
value="permit" validation="" mandatory="no"/>
<vnsParamInst name="order1" locked="no" key="order" cardinality="unspecified" value="10"
validation="" mandatory="no"/>

<vnsFolderInst name="dest-service" nodeNameOrLbl="N1" graphNameOrLbl="FW-Graph"
ctrctNameOrLbl="Client-to-Web" scopedBy="epg" locked="no" key="destination_service"
devCtxLbl="" cardinality="unspecified">
<vnsParamInst name="op" locked="no" key="operator" cardinality="unspecified"
value="eq" validation="" mandatory="no"/>
<vnsParamInst name="port" locked="no" key="low_port" cardinality="unspecified"
value="https" validation="" mandatory="no"/>
</vnsFolderInst>
<vnsFolderInst name="dest-address" nodeNameOrLbl="N1" graphNameOrLbl="FW-Graph"
ctrctNameOrLbl="Client-to-Web" scopedBy="epg" locked="no" key="destination_address"
devCtxLbl="" cardinality="unspecified">
<vnsParamInst name="any" locked="no" key="any" cardinality="unspecified" value="any"
validation="" mandatory="no"/>
</vnsFolderInst>

<vnsFolderInst name="src-address" nodeNameOrLbl="N1" graphNameOrLbl="FW-Graph"
ctrctNameOrLbl="Client-to-Web" scopedBy="epg" locked="no" key="source_address"
devCtxLbl="" cardinality="unspecified">
<vnsParamInst name="any" locked="no" key="any" cardinality="unspecified" value="any"
validation="" mandatory="no"/>
</vnsFolderInst>

<vnsFolderInst name="tcp" nodeNameOrLbl="N1" graphNameOrLbl="FW-Graph"
ctrctNameOrLbl="Client-to-Web" scopedBy="epg" locked="no" key="protocol" devCtxLbl=""
cardinality="unspecified">
<vnsParamInst name="tcp" locked="no" key="name_number" cardinality="unspecified"
value="tcp" validation="" mandatory="no"/>
</vnsFolderInst>
</vnsFolderInst>

```

```

<vnsFolderInst name="permit-http" nodeNameOrLbl="N1" graphNameOrLbl="FW-Graph"
ctrctNameOrLbl="Client-to-Web" scopedBy="epg" locked="no" key="AccessControlEntry"
devCtxLbl="" cardinality="unspecified">
<vnsParamInst name="action-permit" locked="no" key="action" cardinality="unspecified"
value="permit" validation="" mandatory="no"/>
<vnsParamInst name="order1" locked="no" key="order" cardinality="unspecified" value="10"
validation="" mandatory="no"/>

<vnsFolderInst name="dest-service" nodeNameOrLbl="N1" graphNameOrLbl="FW-Graph"
ctrctNameOrLbl="Client-to-Web" scopedBy="epg" locked="no" key="destination_service"
devCtxLbl="" cardinality="unspecified">
<vnsParamInst name="op" locked="no" key="operator" cardinality="unspecified" value="eq"
validation="" mandatory="no"/>
<vnsParamInst name="port" locked="no" key="low_port" cardinality="unspecified" value="http"
validation="" mandatory="no"/>
</vnsFolderInst>

<vnsFolderInst name="dest-address" nodeNameOrLbl="N1" graphNameOrLbl="FW-Graph"
ctrctNameOrLbl="Client-to-Web" scopedBy="epg" locked="no" key="destination_address"
devCtxLbl=""
cardinality="unspecified">
<vnsParamInst name="any" locked="no" key="any" cardinality="unspecified" value="any"
validation="" mandatory="no"/>
</vnsFolderInst>

<vnsFolderInst name="src-address" nodeNameOrLbl="N1" graphNameOrLbl="FW-Graph"
ctrctNameOrLbl="Client-to-Web" scopedBy="epg" locked="no" key="source_address" devCtxLbl=""
cardinality="unspecified">
<vnsParamInst name="any" locked="no" key="any" cardinality="unspecified" value="any"
validation="" mandatory="no"/>
</vnsFolderInst>

<vnsFolderInst name="tcp" nodeNameOrLbl="N1" graphNameOrLbl="FW-Graph"
ctrctNameOrLbl="Client-to-Web" scopedBy="epg" locked="no" key="protocol" devCtxLbl=""
cardinality="unspecified">
<vnsParamInst name="tcp" locked="no" key="name_number" cardinality="unspecified" value="tcp"
validation="" mandatory="no"/>
</vnsFolderInst>
</vnsFolderInst>
</vnsFolderInst>
<fvRsProv prio="unspecified" matchT="AtleastOne" tnVzBrCPName="Client-to-Web"/>
</fvAEPg>

<fvAEPg name="Client" descr="" prio="unspecified" pcEnfPref="unenforced" matchT="AtleastOne"
isAttrBasedEPg="no" fwdCtrl="">
<fvRsCons prio="unspecified" tnVzBrCPName="Client-to-Web"/>
<fvRsDomAtt tDn="uni/vmmp-VMware/dom-ACI_vDS" resImedcy="lazy" primaryEncap="unknown"
instrImedcy="lazy" encap="unknown" delimiter="" classPref="encap"/>
<fvRsBd tnFvBDName="BD1"/>
<fvRsCustQosPol tnQosCustomPolName=""/>
</fvAEPg>
</fvAp>
<fvRsTenantMonPol tnMonEPGPName=""/>

<vnsLDevVip name="ASAv" managed="yes" isCopy="no" funcType="GoTo" trunking="no"
svcType="FW" packageModel="ASAv" mode="legacy-Mode" devtype="VIRTUAL"
contextAware="single-Context">
<vnsCCred name="username" value="admin"/>

```

```

<vnsRsMDevAtt tDn="uni/infra/mDev-CISCO-ASA-1.2"/>
<vnsCCredSecret name="password"/>
<vnsCMgmt name="" port="443" host="172.31.184.249"/>
<vnsRsALDevToDomP tDn="uni/vmmp-VMware/dom-ACI_vDS"/>

<vnsCDev name="Device1" devCtxLbl="" vmName="ASAv-L3" vcenterName="vcenter">
<vnsCCred name="username" value="admin"/>
<vnsCCredSecret name="password"/>
<vnsCMgmt name="" port="443" host="172.31.184.249"/>
<vnsCIf name="GigabitEthernet0/1" vnicName="Network adapter 3"/>
<vnsCIf name="GigabitEthernet0/0" vnicName="Network adapter 2"/>
<vnsRsCDevToCtrlrP tDn="uni/vmmp-VMware/dom-ACI_vDS/ctrlr-vcenter"/>
</vnsCDev>

<vnsLIf name="provider" encap="unknown">
<vnsRsMetaIf tDn="uni/infra/mDev-CISCO-ASA-1.2/mIfLbl-internal" isConAndProv="no"/>
<vnsRsCIfAttN tDn="uni/tn-Tenant-test/1DevVip-ASAv/cDev-Device1/cIf-[GigabitEthernet0/1]"/>
</vnsLIf>

<vnsLIf name="consumer" encap="unknown">
<vnsRsMetaIf tDn="uni/infra/mDev-CISCO-ASA-1.2/mIfLbl-external" isConAndProv="no"/>
<vnsRsCIfAttN tDn="uni/tn-Tenant-test/1DevVip-ASAv/cDev-Device1/cIf-[GigabitEthernet0/0]"/>
</vnsLIf>
</vnsLDevVip>
</fvTenant>

```

Example: Configuring Layer 4 to Layer 7 Route Peering

Configuring Layer 4 to Layer 7 Route Peering With the REST API

These `l3extOut` policies specify the OSPF configurations needed to enable OSPF on the fabric leaf and are very similar to the `l3extOut` policies used for external communication.

The `l3extOut` policies also specify the prefix-based EPGs that control which routes are distributed in/out of the fabric. The `scope=import` attribute controls two things: which endpoint prefixes are learned; and directs the external L4-L7 device to advertise this route. The `scope=export` attribute specifies that the fabric has to advertise this route to the L4-L7 device.

Two sample `l3extOut` policies are shown below: `OspfInternal` deployed on `eth1/23`, and `OspfExternal` deployed on `eth1/25`.

Before you begin

Create one or more `l3extOut` external network connections and deploy them on the fabric leaf nodes where the service device is connected.

Step 1 To configure `OspfInternal` on `eth1/23`, send a post with XML similar to the following example:

Example:

```

<?xml version="1.0" encoding="UTF-8?>
<!-- /api/policymgr/mo.xml -->
<polUni>
  <fvTenant name="coke{{tenantId}}">
    {% if status is not defined %}
      {% set status = "created,modified" %}

```

```

{% endif %}

<l3extOut name="OspfInternal" status="{{status}}">
  <l3extLNodeP name="bLeaf-101">
    <l3extRsNodeL3OutAtt tDn="topology/pod-1/node-101" rtrId="180.0.0.11"/>
    <l3extLIfP name='portIf'>
      <l3extRsPathL3OutAtt tDn="topology/pod-1/paths-101/pathep-[eth1/23]"
        ifInstT='ext-svi' encap='vlan-3844' addr="30.30.30.100/28" mtu='1500' />
      <!-- <ospfIfP authKey="tecom" authType="md5" authKeyId='1'> -->
      <ospfIfP>
        <ospfRsIfPol tnOspfIfPolName='ospfIfPol' />
      </ospfIfP>
    </l3extLIfP>
  </l3extLNodeP>

  <ospfExtP areaId='111' areaType='nssa' areaCtrl='redistribute' />

  <l3extInstP name="OspfInternalInstP">
    <l3extSubnet ip="30.30.30.100/28" scope="import" />
    <l3extSubnet ip="20.20.20.0/24" scope="import" />
    <l3extSubnet ip="10.10.10.0/24" scope="export" />
  </l3extInstP>

  <l3extRsEctx tnFvCtxName="cokectx1" />

</l3extOut>

<ospfIfPol name="ospfIfPol" nwT='bcast' xmitDelay='1'
  helloIntvl='10' deadIntvl='40' status="created,modified" />
</fvTenant>
</polUni>

```

Step 2 To configure OspfExternal on eth1/25, send a post with XML similar to the following example:

Example:

```

<?xml version="1.0" encoding="UTF-8?>
<!-- /api/policymgr/mo.xml -->
<polUni>
  <fvTenant name="common">
    <fvCtx name="commonctx" />

    {% if status is not defined %}
      {% set status="created,modified" %}
    {% endif %}

  <l3extOut name="OspfExternal" status="{{status}}">
    <l3extLNodeP name="bLeaf-101">
      <l3extRsNodeL3OutAtt tDn="topology/pod-1/node-101" rtrId="180.0.0.8/28" />
      <l3extLIfP name='portIf'>
        {% if intfType is not defined %}
          {% set intfType="ext-svi" %}
        {% endif %}
      <l3extRsPathL3OutAtt tDn="topology/pod-1/paths-101/pathep-[eth1/25]"
        ifInstT='ext-svi' encap='vlan-3843' addr="40.40.40.100/28" mtu='1500' />
      <!-- ospfIfP authKey="tecom" authType="md5" authKeyId='1'> -->
      <ospfIfP>
        <ospfRsIfPol tnOspfIfPolName='ospfIfPol' />
      </ospfIfP>
    </l3extLIfP>
  </l3extLNodeP>

```

```

<ospfExtP areaId='111' areaType='nssa' areaCtrl='redistribute' />

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

  <l3extRsEctx tnfvCtxName="commonctx" />

</l3extOut>

<ospfIfPol name="ospfIfPol" nwT='bcast' xmitDelay='1' helloIntvl='10' deadIntvl='40'
status="created,modified" />
</fvTenant>
</polUni>

```

The `l3extInstP` object specifies that prefixes 40.40.40.100/28 and 10.10.10.0/24 are to be used for prefix based endpoint association and indicate that the L4-L7 device should advertise these routes.

The `l3extRsPathL3OutAtt` object specifies where each L3extOut is deployed.

Note For route peering to work, the `l3extRsPathL3OutAtt` must match the `RsCifPathAtt` where the L4-L7 logical device cluster is connected.

Specifying an L3extOut Policy for Layer 4 to L7 Route Peering

A specific `l3extOut` policy can be used for a logical device cluster using its selection policy `vnsLifCtx`. The `vnsRsLifCtxToInstP` points the `LifCtx` to the appropriate **OspfInternal** and **OspfExternal** `l3extInstP` EPGs. To configure an L3extOut policy used for Layer 4 to Layer 7 Route Peering, send a post with XML such as the following example:

```

<vnsLDevCtx ctrctNameOrLbl="webCtrct{{graphId}}" graphNameOrLbl="WebGraph" nodeNameOrLbl="FW">

  <vnsRsLDevCtxToLDev tDn="uni/tn-solar{{tenantId}}/lDevVip-Firewall" />
  <vnsLifCtx connNameOrLbl="internal">
    {% if L3ExtOutInternal is not defined %}
    <fvSubnet ip="10.10.10.10/24" />
    {% endif %}
    <vnsRsLifCtxToBD tDn="uni/tn-solar{{tenantId}}/BD-solarBD1" />
    <vnsRsLifCtxToLif tDn="uni/tn-solar{{tenantId}}/lDevVip-Firewall/lif-internal" />
    {% if L3ExtOutInternal is defined %}
    <vnsRsLifCtxToInstP
tDn="uni/tn-solar{{tenantId}}/out-OspfInternal/instP-OspfInternalInstP"
status={{L3ExtOutInternal}}"/>
    {% endif %}
  </vnsLifCtx>
  <vnsLifCtx connNameOrLbl="external">
    {% if L3ExtOutExternal is not defined %}
    <fvSubnet ip="40.40.40.40/24" />
    {% endif %}
    <vnsRsLifCtxToBD tDn="uni/tn-solar{{tenantId}}/BD-solarBD4" />
    <vnsRsLifCtxToLif tDn="uni/tn-solar{{tenantId}}/lDevVip-Firewall/lif-external" />
    {% if L3ExtOutExternal is defined %}
    <vnsRsLifCtxToInstP
tDn="uni/tn-solar{{tenantId}}/out-OspfExternal/instP-OspfExternalInstP"
status={{L3ExtOutExternal}}"/>
    {% endif %}
  </vnsLifCtx>
</vnsLDevCtx>

```



```

    {% endif %}
  </vnsLIfCtx>
</vnsLDevCtx>

```

The associated concrete device needs to have a `vnsRsCifPathAtt` that deploys it to the same fabric leaf as shown in the following example:

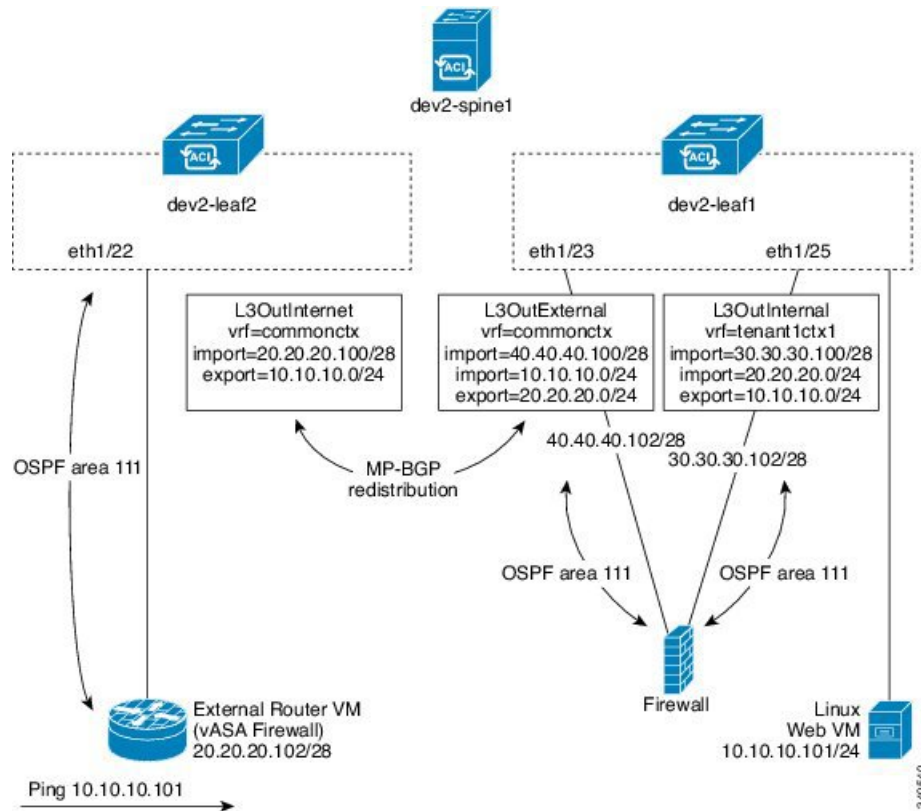
```

<vnsCDev name="ASA">
  <vnsRsLDevCtxToLDev tDn="uni/tn-solar{{tenantId}}/lDevVip-Firewall"/>
  <vnsCIf name="Gig0/0">
    <vnsRsCifPathAtt tDn="topology/pod-1/paths-101/pathep-[eht1/23]"/>
  </vnsCIf>
  <vnsCIf name="Gig0/1">
    <vnsRsCifPathAtt tDn="topology/pod-1/paths-101/pathep-[eht1/25]"/>
  </vnsCIf>
  <vnsCMgmt name="devMgmt" host="{{asaIp}}" port="443" />
  <vnsCCred name="username" value="admin" />
  <vnsCCredSecret name="password" value="insieme" />
</vnsCDev>

```

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

Figure 52: Sample Deployment



In this 2-leaf, 1-spine topology, the linux web server is at IP 10.10.10.101/24 and is hosted on an ESX server connected to dev2-leaf1. A service graph is deployed consisting of a two-arm firewall that is also connected to dev2-leaf1. The service graph is associated with a contract that binds an external `l3extOut` **L3OutInternet** with the provider EPG (Web VM). Two internal `l3extOut` policies, an **L3OutExternal**, and an **L3OutInternal** are also deployed to the leaf ports where the service device is connected.



CHAPTER 18

Configuring Security

- [Enabling TACACS+, RADIUS, and LDAP, on page 457](#)
- [Configuring FIPS, on page 460](#)
- [Configuring Fabric Secure Mode, on page 461](#)
- [Enabling RBAC, on page 462](#)
- [Enabling Port Security, on page 476](#)
- [Enabling COOP Authentication, on page 478](#)
- [Enabling Control Plane Policing, on page 479](#)
- [Configuring First Hop Security, on page 484](#)
- [Configuring 802.1x, on page 487](#)

Enabling TACACS+, RADIUS, and LDAP

Overview

This article provides step by step instructions on how to enable RADIUS, TACACS+, and LDAP users to access the APIC. It assumes the reader is thoroughly familiar with the *Cisco Application Centric Infrastructure Fundamentals* manual, especially the User Access, Authentication, and Accounting chapter.



Note In the case of a disaster scenario such as the loss of all but one APIC in the cluster, APIC disables remote authentication. In this scenario, only a local administrator account can log into the fabric devices.



Note Remote users for AAA Authentication with `shell:domains=all/read-all/` will not be able to access Leaf switches and Spine switches in the fabric for security purposes. This pertains to all version up to 4.0(1h).

Configuring APIC for TACACS+ Using the REST API

- The Cisco Application Centric Infrastructure (ACI) fabric must be installed, Application Policy Infrastructure Controllers (APICs) must be online, and the APIC cluster must be formed and healthy.

- The TACACS+ server host name or IP address, port, and key must be available.
- The APIC management endpoint group must be available.

Step 1 Configure the TACACS+ Provider by sending a POST request with XML such as the following example:

Example:

```
<aaaTacacsPlusProvider timeout="5" retries="1" port="49" name="192.168.200.1" monitoringUser="test"
  monitorServer="disabled"
  dn="uni/userext/tacacsxt/tacacsplusprovider-192.168.200.1" authProtocol="pap"/>
```

Step 2 Configure the TACACS+ Provider Group by sending a POST request with XML such as the following example:

Example:

```
<aaaTacacsPlusProviderGroup name="TENANT64_TACACS_provGrp"
  dn="uni/userext/tacacsxt/tacacsplusprovidergroup-TENANT64_TACACS_provGrp"/>
```

Step 3 Configure the TACACS+ Login Domain by sending a POST request with XML such as the following example:

Example:

```
<aaaLoginDomain name="TENANT64_TACACS_LoginDom" dn="uni/userext/logindomain-TENANT64_TACACS_LoginDom"/>
```

Example

The entire configuration can be sent in one POST request, with XML such as this example:

```
<aaaTacacsPlusProvider timeout="5" retries="1" port="49"
  name="192.168.200.1" monitoringUser="test" monitorServer="disabled"
  dn="uni/userext/tacacsxt/tacacsplusprovider-192.168.200.1" authProtocol="pap"/>
<aaaTacacsPlusProviderGroup name="TENANT64_TACACS_provGrp"
  dn="uni/userext/tacacsxt/tacacsplusprovidergroup-TENANT64_TACACS_provGrp"/>
<aaaLoginDomain name="TENANT64_TACACS_LoginDom"
  dn="uni/userext/logindomain-TENANT64_TACACS_LoginDom"/>
```

Configuring APIC for RADIUS Using the REST API

Before you begin

- The ACI fabric must be installed, Application Policy Infrastructure Controllers (APICs) must be online, and the APIC cluster must be formed and healthy.
- The RADIUS server host name or IP address, port, authorization protocol, and key must be available.
- The APIC management endpoint group must be available.

Step 1 Configure the RADIUS Provider by sending a POST request with XML such as the following example:

Example:

```
<aaaRadiusProvider timeout="5" retries="1" name="TENANT64_RADIUS-host.com"
  monitoringUser="test" monitorServer="disabled"
  dn="uni/userext/radiusxt/radiusprovider-TENANT64_RADIUS-host.com" authProtocol="pap" authPort="1812"/>
```

Step 2 Configure the RADIUS Provider Group by sending a POST request with XML such as the following example:

Example:

```
<aaaRadiusProviderGroup name="TENANT64_RADIUS_provGrp"
dn="uni/userext/radiusext/radiusprovidergroup-TENANT64_RADIUS_provGrp"/>
```

Step 3 Configure the RADIUS Login Domain by sending a POST request with XML such as the following example:

Example:

```
<aaaLoginDomain name="TENANT64_RADIUSLoginDom"
dn="uni/userext/logindomain-TENANT64_RADIUSLoginDom"/>
```

Example

The entire configuration can be sent as one POST request, with XML such as this example:

```
<aaaRadiusProvider
timeout="5" retries="1" name="TENANT64_RADIUS-host.com" monitoringUser="test"
monitorServer="disabled"
dn="uni/userext/radiusext/radiusprovider-TENANT64_RADIUS-host.com" authProtocol="pap"
authPort="1812"/>
<aaaRadiusProviderGroup
name="TENANT64_RADIUS_provGrp"
dn="uni/userext/radiusext/radiusprovidergroup-TENANT64_RADIUS_provGrp"/>
<aaaLoginDomain
name="TENANT64_RADIUSLoginDom" dn="uni/userext/logindomain-TENANT64_RADIUSLoginDom"/>
```

Configuring APIC for LDAP Using the REST API

Before you begin

- The Cisco Application Centric Infrastructure (ACI) fabric must be installed, Application Policy Infrastructure Controllers (APICs) must be online, and the APIC cluster must be formed and healthy.
- The LDAP server host name or IP address, port, bind DN, Base DN, and password must be available.
- The APIC management endpoint group must be available.

Step 1 Configure the LDAP Provider by sending a POST request with XML such as the following example:

Example:

```
<aaaLdapProvider timeout="30" rootdn="" retries="1" port="389" name="TENANT64_LDAP-host.com"
monitoringUser="test" monitorServer="disabled" filter="cn=$userid" enableSSL="yes"
dn="uni/userext/ldapext/ldaprovider-TENANT64_LDAP-host.com" descr="" basedn=""
attribute="CiscoAVPair" SSLValidationLevel="strict"/>
```

Step 2 Configure the LDAP Provider Group by sending a POST request with XML such as the following example:

Example:

```
<aaaLdapProviderGroup name="TENANT64_LDAP-ProvGrp"
dn="uni/userext/ldapext/ldaprovidergroup-TENANT64_LDAP-ProvGrp"/>
```

Step 3 Configure the LDAP Login Domain by sending a POST request with XML such as the following example:

Example:

```
<aaaDomainAuth realm="ldap" providerGroup="TENANT64_LDAP-ProvGrp"
dn="uni/userext/logindomain-TENANT64_LDAPLoginDom/domainauth"/>
```

Example

The entire configuration can be sent in one POST request, with XML such as the following example:

```
<aaaLdapProvider
timeout="30" rootdn="" retries="1" port="389" name="TENANT64_LDAP-host.com"
monitoringUser="test" monitorServer="disabled" filter="cn=$userid" enableSSL="yes"
dn="uni/userext/ldapext/ldaprovider-TENANT64_LDAP-host.com" descr="" basedn=""
attribute="CiscoAVPair" SSLValidationLevel="strict"/>
<aaaLdapProviderGroup
name="TENANT64_LDAP-ProvGrp"dn="uni/userext/ldapext/ldaprovidergroup-TENANT64_LDAP-ProvGrp"/>
<aaaDomainAuth
realm="ldap" providerGroup="TENANT64_LDAP-ProvGrp"
dn="uni/userext/logindomain-TENANT64_LDAPLoginDom/domainauth"/>
```

Configuring FIPS

About Federal Information Processing Standards (FIPS)

The Federal Information Processing Standards (FIPS) Publication 140-2, Security Requirements for Cryptographic Modules, details the U.S. government requirements for cryptographic modules. FIPS 140-2 specifies that a cryptographic module should be a set of hardware, software, firmware, or some combination that implements cryptographic functions or processes, including cryptographic algorithms and, optionally, key generation, and is contained within a defined cryptographic boundary.

FIPS specifies certain cryptographic algorithms as secure, and it also identifies which algorithms should be used if a cryptographic module is to be called FIPS compliant.

Guidelines and Limitations for FIPS

The following guidelines and limitations apply to FIPS:

- When FIPS is enabled, FIPS is applied across the Cisco Application Policy Infrastructure Controller (APIC).
- When FIPS is enabled, you must disable FIPS before you downgrade the Cisco APIC to a release that does not support FIPS.
- Make your passwords a minimum of eight characters in length.
- Disable Telnet. Log in using only SSH.
- Delete all SSH Server RSA1 keypairs.
- Secure Shell (SSH) and SNMP are supported.

- Disable SNMP v1 and v2. Any existing user accounts on the switch that have been configured for SNMPv3 should be configured only with SHA for authentication and AES for privacy.
- Disable remote authentication through RADIUS/TACACS+. Only local and LDAP users can be authenticated.
- After enabling FIPS on the Cisco APIC, reload the dual supervisor spine switches twice for FIPS to take effect.
- On a dual supervisor spine switch that has FIPS enabled, if a supervisor is replaced, then the spine switch must be reloaded twice for FIPS to take effect on the new supervisor.
- Starting with the 2.3(1) release, FIPS can be configured at the switch level.
- Starting with the 3.1(1) release, when FIPS is enabled, NTP will operate in FIPS mode, Under FIPS mode NTP supports authentication with HMAC-SHA1 and no authentication.

Configuring FIPS for Cisco APIC Using REST API

When FIPS is enabled, it is applied across Cisco APIC.

Configure FIPS for all tenants.

Example:

```
https://apic1.cisco.com/api/node/mo/uni/userext.xml
<aaaFabricSec fipsMode="enable" />
```

Note You must reboot to complete the configuration. Anytime you change the mode, you must reboot to complete the configuration.

Configuring Fabric Secure Mode

Fabric Secure Mode

Fabric secure mode prevents parties with physical access to the fabric equipment from adding a switch or APIC controller to the fabric without manual authorization by an administrator. Starting with release 1.2(1x), the firmware checks that switches and controllers in the fabric have valid serial numbers associated with a valid Cisco digitally signed certificate. This validation is performed upon upgrade to this release or during an initial installation of the fabric. The default setting for this feature is permissive mode; an existing fabric continues to run as it has after an upgrade to release 1.2(1) or later. An administrator with fabric-wide access rights must enable strict mode. The following table summarizes the two modes of operation:

Permissive Mode (default)	Strict Mode
Allows an existing fabric to operate normally even though one or more switches have an invalid certificate.	Only switches with a valid Cisco serial number and SSL certificate are allowed.

Permissive Mode (default)	Strict Mode
Does not enforce serial number based authorization .	Enforces serial number authorization.
Allows auto-discovered controllers and switches to join the fabric without enforcing serial number authorization.	Requires an administrator to manually authorize controllers and switches to join the fabric.

Configuring Fabric Secure Mode Using the REST API

To manage Secure Fabric Mode using the REST API, perform the following steps:

Step 1 To enable strict mode, send a POST request with XML such as the following example:

Example:

```
POST https://apic-ip-address/api/node/mo/uni.xml?
<pkiFabricCommunicationEp mode="strict"/>
```

Step 2 To enable permissive mode, send a POST request with XML such as the following example:

Example:

```
POST https://apic-ip-address/api/node/mo/uni.xml?
<pkiFabricCommunicationEp mode="permissive"/>
```

Step 3 To authorize a controller, send a POST request with XML such as the following example:

Example:

```
POST https://apic-ip-address/api/mo/uni/controller.xml?
<fabricNodeIdentPol>
  <fabricCtrlrIdentP serial="TEP-1-1"/>
</fabricNodeIdentPol>
```

Step 4 To reject a controller, send a POST request with XML such as the following example:

Example:

```
POST https://apic-ip-address/api/mo/uni/controller.xml?
<fabricNodeIdentPol>
  <fabricCtrlrIdentP serial="FCH1750V025" reject="yes"/>
</fabricNodeIdentPol>
```

Enabling RBAC

Access Rights Workflow Dependencies

The Cisco Application Centric Infrastructure (ACI) RBAC rules enable or restrict access to some or all of the fabric. For example, in order to configure a leaf switch for bare metal server access, the logged in administrator must have rights to the `infra` domain. By default, a tenant administrator does not have rights to the `infra` domain. In this case, a tenant administrator who plans to use a bare metal server connected to a leaf switch

could not complete all the necessary steps to do so. The tenant administrator would have to coordinate with a fabric administrator who has rights to the `infra` domain. The fabric administrator would set up the switch configuration policies that the tenant administrator would use to deploy an application policy that uses the bare metal server attached to an ACI leaf switch.

AAA RBAC Roles and Privileges

The Application Policy Infrastructure Controller (APIC) provides the following AAA roles and privileges:



Note For each of the defined roles in Cisco APIC, the *APIC Roles and Privileges Matrix* shows which managed object classes can be written and which can be read. The matrix can be found at this URL: <https://www.cisco.com/c/dam/en/us/td/docs/Website/datacenter/apicroles/roles.html>

Role	Privilege	Description
aaa	aaa	Used for configuring authentication, authorization, accounting, and import/export policies.
admin	admin	Provides full access to all of the features of the fabric. The admin privilege can be considered to be a union of all other privileges.

Role: access-admin

Privilege	Description
access-connectivity-l1	Used for Layer 1 configuration under <code>infra</code> . Example: selectors and port Layer 1 policy configurations.
access-connectivity-l2	Used for Layer 2 configuration under <code>infra</code> . Example: encapsulations on selectors, and attachable entity.
access-connectivity-l3	Used for Layer 3 configuration under <code>infra</code> and static route configurations under a tenant's L3Out.
access-connectivity-mgmt	Used for management <code>infra</code> policies.
access-connectivity-util	Used for tenant ERSPAN policies.
access-equipment	Used for access port configuration.
access-protocol-l1	Used for Layer 1 protocol configurations under <code>infra</code> .
access-protocol-l2	Used for Layer 2 protocol configurations under <code>infra</code> .
access-protocol-l3	Used for Layer 3 protocol configurations under <code>infra</code> .
access-protocol-mgmt	Used for fabric-wide policies for NTP, SNMP, DNS, and image management.

Role: access-admin	
Privilege	Description
access-protocol-ops	Used for operations-related access policies such as cluster policy and firmware policies.
access-qos	Used for changing CoPP and QoS-related policies.
Role: fabric-admin	
Privilege	Description
fabric-connectivity-l1	Used for Layer 1 configuration under the fabric. Example: selectors and port Layer 1 policy and vPC protection.
fabric-connectivity-l2	Used in firmware and deployment policies for raising warnings for estimating policy deployment impact.
fabric-connectivity-l3	Used for Layer 3 configuration under the fabric. Example: Fabric IPv4, IPv6, and MAC protection groups.
fabric-connectivity-mgmt	Used for atomic counter and diagnostic policies on leaf switches and spine switches.
fabric-connectivity-util	Used for atomic counter, diagnostic, and image management policies on leaf switches and spine switches.
fabric-equipment	Used for atomic counter, diagnostic, and image management policies on leaf switches and spine switches.
fabric-protocol-l1	Used for Layer 1 protocol configurations under the fabric.
fabric-protocol-l2	Used for Layer 2 protocol configurations under the fabric.
fabric-protocol-l3	Used for Layer 3 protocol configurations under the fabric.
fabric-protocol-mgmt	Used for fabric-wide policies for NTP, SNMP, DNS, and image management.
fabric-protocol-ops	Used for ERSPAN and health score policies.
fabric-protocol-util	Used for firmware management traceroute and endpoint tracking policies.
tenant-connectivity-util	Used for atomic counter, diagnostic, and image management policies on leaf switches and spine switches.
tenant-connectivity-l2	Used for Layer 2 connectivity changes, including bridge domains and subnets.
tenant-connectivity-l3	Used for Layer 3 connectivity changes, including VRFs.
tenant-protocol-ops	Used for tenant traceroute policies.

Role	Privilege	Description
nw-svc-admin	nw-svc-device	Used for managing Layer 4 to Layer 7 service devices.
	nw-svc-devshare	Used for managing shared Layer 4 to Layer 7 service devices.
	nw-svc-policy	Used for managing Layer 4 to Layer 7 network service orchestration.
nw-svc-params	nw-svc-params	Used for managing Layer 4 to Layer 7 service policies.

Role: ops	
Privilege	Description
ops	<p>Used for viewing the policies configured including troubleshooting policies.</p> <p>Note The ops role cannot be used for creating new monitoring and troubleshooting policies. Those policies need to be created by using the admin privilege, just like any other configurations in the Cisco APIC.</p>

Role: read-all	
Privilege	Description
access-connectivity-l1	Used for Layer 1 configuration under infra. Example: selectors and port Layer 1 policy configurations.
access-connectivity-l2	Used for Layer 2 configuration under infra. Example: Encap configurations on selectors, and attachable entity.
access-connectivity-l3	Used for Layer 3 configuration under infra and static route configurations under a tenant's L3Out.
access-connectivity-mgmt	Used for management infra policies.
access-connectivity-util	Used for tenant ERSPAN policies.
access-equipment	Used for access port configuration.
access-protocol-l1	Used for Layer 1 protocol configurations under infra.
access-protocol-l2	Used for Layer 2 protocol configurations under infra.
access-protocol-l3	Used for Layer 3 protocol configurations under infra.
access-protocol-mgmt	Used for fabric-wide policies for NTP, SNMP, DNS, and image management.

Role: read-all	
Privilege	Description
access-protocol-ops	Used for operations-related access policies such as cluster policy and firmware policies.
access-qos	Used for changing CoPP and QoS-related policies.
fabric-connectivity-l1	Used for Layer 1 configuration under the fabric. Example: selectors and port Layer 1 policy and vPC protection.
fabric-connectivity-l2	Used in firmware and deployment policies for raising warnings for estimating policy deployment impact.
fabric-connectivity-l3	Used for Layer 3 configuration under the fabric. Example: Fabric IPv4, IPv6, and MAC protection groups.
fabric-protocol-l1	Used for Layer 1 protocol configurations under the fabric.
fabric-protocol-l2	Used for Layer 2 protocol configurations under the fabric.
fabric-protocol-l3	Used for Layer 3 protocol configurations under the fabric.
nw-svc-device	Used for managing Layer 4 to Layer 7 service devices.
nw-svc-devshare	Used for managing shared Layer 4 to Layer 7 service devices.
nw-svc-params	Used for managing Layer 4 to Layer 7 service policies.
nw-svc-policy	Used for managing Layer 4 to Layer 7 network service orchestration.
ops	Used for viewing the policies configured including troubleshooting policies. Note The ops role cannot be used for creating new monitoring and troubleshooting policies. Those policies need to be created by using the admin privilege, just like any other configurations in the Cisco APIC.
tenant-connectivity-util	Used for atomic counter, diagnostic, and image management policies on leaf switches and spine switches.
tenant-connectivity-l2	Used for Layer 2 connectivity changes, including bridge domains and subnets.
tenant-connectivity-l3	Used for Layer 3 connectivity changes, including VRFs.
tenant-connectivity-mgmt	Used for tenant in-band and out-of-band management connectivity configurations and for debugging/monitoring policies such as atomic counters and health score.
tenant-epg	Used for managing tenant configurations such as deleting/creating endpoint groups.

Role: read-all	
Privilege	Description
tenant-ext-connectivity-l1	Used for write access firmware policies.
tenant-ext-connectivity-l2	Used for managing tenant L2Out configurations.
tenant-ext-connectivity-l3	Used for managing tenant L3Out configurations.
tenant-ext-connectivity-mgmt	Used as write access for firmware policies.
tenant-ext-connectivity-util	Used for debugging/monitoring/observer policies such as traceroute, ping, oam, and eptrk.
tenant-ext-protocol-l1	Used for managing tenant external Layer 1 protocols. Generally only used for write access for firmware policies.
tenant-ext-protocol-l2	Used for managing tenant external Layer 2 protocols. Generally only used for write access for firmware policies.
tenant-ext-protocol-l3	Used for managing tenant external Layer 3 protocols such as BGP, OSPF, PIM, and IGMP.
tenant-ext-protocol-mgmt	Used as write access for firmware policies.
tenant-ext-protocol-util	Used for debugging/monitoring/observer policies such as traceroute, ping, oam, and eptrk.
tenant-network-profile	Used for managing tenant configurations, such as deleting and creating network profiles, and deleting and creating endpoint groups.
tenant-protocol-l1	Used for managing configurations for Layer 1 protocols under a tenant.
tenant-protocol-l2	Used for managing configurations for Layer 2 protocols under a tenant.
tenant-protocol-l3	Used for managing configurations for Layer 3 protocols under a tenant.
tenant-protocol-mgmt	Only used as write access for firmware policies.
tenant-protocol-ops	Used for tenant traceroute policies.
tenant-QoS	Used for QoS-related configurations for a tenant.
tenant-security	Used for contract-related configurations for a tenant.
vmm-connectivity	Used to read all the objects in Cisco APIC's VMM inventory required for virtual machine connectivity.
vmm-ep	Used to read virtual machine and hypervisor endpoints in the Cisco APIC's VMM inventory.
vmm-policy	Used for managing policies for virtual machine networking.
vmm-protocol-ops	Not used by VMM policies.

Role: read-all	
Privilege	Description
vmm-security	Used for managing authentication policies for VMM, such as the username and password for VMware vCenter.
Role: tenant-admin	
Privilege	Description
aaa	Used for configuring authentication, authorization, accounting and import/export policies.
access-connectivity-l1	Used for Layer 1 configuration under infra. Example: selectors and port Layer 1 policy configurations.
access-connectivity-l2	Used for Layer 2 configuration under infra. Example: Encap configurations on selectors, and attachable entity.
access-connectivity-l3	Used for Layer 3 configuration under infra and static route configurations under a tenant's L3Out.
access-connectivity-mgmt	Used for management infra policies.
access-connectivity-util	Used for tenant ERSPAN policies.
access-equipment	Used for access port configuration.
access-protocol-l1	Used for Layer 1 protocol configurations under infra.
access-protocol-l2	Used for Layer 2 protocol configurations under infra.
access-protocol-l3	Used for Layer 3 protocol configurations under infra.
access-protocol-mgmt	Used for fabric-wide policies for NTP, SNMP, DNS, and image management.
access-protocol-ops	Used for operations-related access policies such as cluster policy and firmware policies.
access-qos	Used for changing CoPP and QoS-related policies.
fabric-connectivity-l1	Used for Layer 1 configuration under the fabric. Example: selectors and port Layer 1 policy and vPC protection.
fabric-connectivity-l2	Used in firmware and deployment policies for raising warnings for estimating policy deployment impact.
fabric-connectivity-l3	Used for Layer 3 configuration under the fabric. Example: Fabric IPv4, IPv6, and MAC protection groups.
fabric-connectivity-mgmt	Used for atomic counter and diagnostic policies on leaf switches and spine switches.

Role: tenant-admin	
Privilege	Description
fabric-connectivity-util	Used for atomic counter, diagnostic, and image management policies on leaf switches and spine switches.
fabric-equipment	Used for atomic counter, diagnostic, and image management policies on leaf switches and spine switches.
fabric-protocol-l1	Used for Layer 1 protocol configurations under the fabric.
fabric-protocol-l2	Used for Layer 2 protocol configurations under the fabric.
fabric-protocol-l3	Used for Layer 3 protocol configurations under the fabric.
fabric-protocol-mgmt	Used for fabric-wide policies for NTP, SNMP, DNS, and image management.
fabric-protocol-ops	Used for ERSPAN and health score policies.
fabric-protocol-util	Used for firmware management traceroute and endpoint tracking policies.
nw-svc-device	Used for managing Layer 4 to Layer 7 service devices.
nw-svc-devshare	Used for managing shared Layer 4 to Layer 7 service devices.
nw-svc-params	Used for managing Layer 4 to Layer 7 service policies.
nw-svc-policy	Used for managing Layer 4 to Layer 7 network service orchestration.
ops	Used for viewing the policies configured including troubleshooting policies. Note The ops role cannot be used for creating new monitoring and troubleshooting policies. Those policies need to be created by using the admin privilege, just like any other configurations in the Cisco APIC.
tenant-connectivity-util	Used for atomic counter, diagnostic, and image management policies on leaf switches and spine switches.
tenant-connectivity-l2	Used for Layer 2 connectivity changes, including bridge domains and subnets.
tenant-connectivity-l3	Used for Layer 3 connectivity changes, including VRFs.
tenant-connectivity-mgmt	Used for tenant in-band and out-of-band management connectivity configurations and for debugging/monitoring policies such as atomic counters and health score.
tenant-epg	Used for managing tenant configurations such as deleting/creating endpoint groups.

Role: tenant-admin	
Privilege	Description
tenant-ext-connectivity-l1	Used for write access firmware policies.
tenant-ext-connectivity-l2	Used for managing tenant L2Out configurations.
tenant-ext-connectivity-l3	Used for managing tenant L3Out configurations.
tenant-ext-connectivity-mgmt	Used as write access for firmware policies.
tenant-ext-connectivity-util	Used for debugging/monitoring/observer policies such as traceroute, ping, oam, and eptrk.
tenant-ext-protocol-l1	Used for managing tenant external Layer 1 protocols. Generally only used for write access for firmware policies.
tenant-ext-protocol-l2	Used for managing tenant external Layer 2 protocols. Generally only used for write access for firmware policies.
tenant-ext-protocol-l3	Used for managing tenant external Layer 3 protocols such as BGP, OSPF, PIM, and IGMP.
tenant-ext-protocol-mgmt	Used as Write access for firmware policies.
tenant-ext-protocol-util	Used for debugging/monitoring/observer policies such as traceroute, ping, oam, and eptrk.
tenant-network-profile	Used for managing tenant configurations, such as deleting and creating network profiles, and deleting and creating endpoint groups.
tenant-protocol-l1	Used for managing configurations for Layer 1 protocols under a tenant.
tenant-protocol-l2	Used for managing configurations for Layer 2 protocols under a tenant.
tenant-protocol-l3	Used for managing configurations for Layer 3 protocols under a tenant.
tenant-protocol-mgmt	Only used as write access for firmware policies.
tenant-protocol-ops	Used for tenant traceroute policies.
tenant-QoS	Used for QoS-related configurations for a tenant.
tenant-security	Used for contract-related configurations for a tenant.
vmm-connectivity	Used to read all the objects in Cisco APIC's VMM inventory required for virtual machine connectivity.
vmm-ep	Used to read virtual machine and hypervisor endpoints in the Cisco APIC's VMM inventory.
vmm-policy	Used for managing policies for virtual machine networking.
vmm-protocol-ops	Not used by VMM policies.

Role: tenant-admin	
Privilege	Description
vmm-security	Used for managing authentication policies for VMM, such as the username and password for VMware vCenter.
Role: tenant-ext-admin	
Privilege	Description
tenant-connectivity-util	Used for atomic counter, diagnostic, and image management policies on leaf switches and spine switches.
tenant-connectivity-l2	Used for Layer 2 connectivity changes, including bridge domains and subnets.
tenant-connectivity-l3	Used for Layer 3 connectivity changes, including VRFs.
tenant-connectivity-mgmt	Used for tenant in-band and out-of-band management connectivity configurations and for debugging/monitoring policies such as atomic counters and health score.
tenant-epg	Used for managing tenant configurations such as deleting/creating endpoint groups.
tenant-ext-connectivity-l1	Used for write access firmware policies.
tenant-ext-connectivity-l2	Used for managing tenant L2Out configurations.
tenant-ext-connectivity-l3	Used for managing tenant L3Out configurations.
tenant-ext-connectivity-mgmt	Used as write access for firmware policies.
tenant-ext-connectivity-util	Used for debugging/monitoring/observer policies such as traceroute, ping, oam, and eptrk.
tenant-ext-protocol-l1	Used for managing tenant external Layer 1 protocols. Generally only used for write access for firmware policies.
tenant-ext-protocol-l2	Used for managing tenant external Layer 2 protocols. Generally only used for write access for firmware policies.
tenant-ext-protocol-l3	Used for managing tenant external Layer 3 protocols such as BGP, OSPF, PIM, and IGMP.
tenant-ext-protocol-mgmt	Used as Write access for firmware policies.
tenant-ext-protocol-util	Used for debugging/monitoring/observer policies such as traceroute, ping, oam, and eptrk.
tenant-network-profile	Used for managing tenant configurations, such as deleting and creating network profiles, and deleting and creating endpoint groups.
tenant-protocol-l1	Used for managing configurations for Layer 1 protocols under a tenant.

Role: tenant-ext-admin	
Privilege	Description
tenant-protocol-l2	Used for managing configurations for Layer 2 protocols under a tenant.
tenant-protocol-l3	Used for managing configurations for Layer 3 protocols under a tenant.
tenant-protocol-mgmt	Only used as write access for firmware policies.
tenant-protocol-ops	Used for tenant traceroute policies.
tenant-QoS	Used for QoS-related configurations for a tenant.
tenant-security	Used for contract-related configurations for a tenant.
vmm-connectivity	Used to read all the objects in Cisco APIC's VMM inventory required for virtual machine connectivity.
vmm-ep	Used to read virtual machine and hypervisor endpoints in the Cisco APIC's VMM inventory.
vmm-policy	Used for managing policies for virtual machine networking.
vmm-protocol-ops	Not used by VMM policies.
vmm-security	Used for managing authentication policies for VMM, such as the username and password for VMware vCenter.

Role: vmm-admin	
Privilege	Description
vmm-connectivity	Used to read all the objects in Cisco APIC's VMM inventory required for virtual machine connectivity.
vmm-ep	Used to read virtual machine and hypervisor endpoints in the Cisco APIC's VMM inventory.
vmm-policy	Used for managing policies for virtual machine networking.
vmm-protocol-ops	Not used by VMM policies.
vmm-security	Used for managing authentication policies for a VMM, such as the username and password for VMware vCenter.

Custom Roles

You can create custom roles and assign privileges to the roles. The interface internally assigns one or more privileges to all managed object classes. In an XML model, privileges are assigned in an access attribute. Privilege bits are assigned at compile time and apply per class, and not per instance or object of the class.

In addition to the 45 privilege bits, the "aaa" privilege bit applies to all AAA-subsystem configuration and read operations. The following table provides a matrix of the supported privilege combinations. The rows in

The table represents Cisco Application Centric Infrastructure (ACI) modules and the columns represent functionality for a given module. A value of "Yes" in a cell indicates that the functionality for the module is accessible and there exists a privilege bit to access that functionality. An empty cell indicates that the particular functionality for module is not accessible by any privilege bit. See the privilege bit descriptions to learn what each bit does.

	Connectivity	OS	Security	Application	Fault	Stats	Provider	Service Profile	Service Chain
VMM	Yes		Yes		Yes	Yes	Yes		
Fabric	Yes	Yes	Yes	Yes	Yes	Yes	Yes		
External	Yes	Yes	Yes		Yes	Yes			Yes
Tenant	Yes	Yes	Yes	EPG, NP	Yes	Yes			Yes
Infra	Yes	Yes	Yes	Yes	Yes	Yes			Yes
Ops					Yes	Yes			
Storage	Yes	Yes	Yes	Yes	Yes	Yes			
Network Service	Yes	Yes	Yes	Yes	Yes	Yes		Yes	

Sample RBAC Rules

The RBAC rules in the sample JSON file below enable both trans-tenant access and tenant access to a VMM domain resource. The resources needed by the consumer are `uni/tn-prov1/brc-webCtrct` and `vmmp-Vmware/dom-Datacenter`.

The following two RBAC rules enable the consumer tenant to post the consumer postman query in the JSON file below.

```
<aaaRbacEp>
  <aaaRbacRule objectDn="uni/vmmp-VMware/dom-Datacenter" domain="cons1"/>
  <aaaRbacRule objectDn="uni/tn-prov1/brc-webCtrct" domain="cons1"/>
</aaaRbacEp>
```

The JSON file below contains these two RBAC rules:

```
{
  "id": "ac62a200-9210-f53b-7114-a8f4cffb9a36",
  "name": "SharedContracts",
  "timestamp": 1398806919868,
  "requests": [
    {
      "collectionId": "ac62a200-9210-f53b-7114-a8f4cffb9a36",
      "id": "2dfc75cc-431e-e136-622c-a577ce7622d8",
      "name": "login as prov1",
      "description": "",
      "url": "http://http://solar.local:8000/api/aaaLogin.json",
      "method": "POST",
      "headers": "",
      "data": {
        "\aaaUser\": {
          "\attributes\": {
            "\name\": "\prov1",
            "\pwd\": "\secret!"
          }
        }
      },
      "dataMode": "raw",
      "timestamp": 0,
      "version": 2,
      "time": 1398807562828
    },
    {
      "collectionId": "ac62a200-9210-f53b-7114-a8f4cffb9a36",
      "id": "56e46db0-77ea-743f-a64e-c5f7b1f59807",
      "name": "Root login",
      "description": "",
      "url": "http://http://solar.local:8000/api/aaaLogin.json",
      "method": "POST",
    }
  ]
}
```

```

"headers": "",
"data":
{"aaaUser": {"attributes": {"name": "admin", "pwd": "secret!"}},
"dataMode": "raw", "timestamp": 0, "responses": [], "version": 2},

{"collectionId": "ac62a200-9210-f53b-7114-a8f4cffb9a36", "id": "804893f1-0915-6d35-169d-3af0eb3e64ec",
"name": "consumer tenant only",
"description": "",
"url": "http://solar.local:8000/api/policymgr/mo/uni/tn-cons1.xml",
"method": "POST",
"headers": "",
"data":
"<fvTenant name=\"cons1\">
  <aaaDomainRef name=\"cons1\"/>\n
</fvTenant>\n",
"dataMode": "raw", "timestamp": 0, "version": 2, "time": 1398968007487},

{"collectionId": "ac62a200-9210-f53b-7114-a8f4cffb9a36", "id": "85802d50-8089-bf8b-4481-f149bec258c8",
"name": "login as cons1",
"description": "",
"url": "http://solar.local:8000/api/aaaLogin.json",
"method": "POST",
"headers": "",
"data":
{"aaaUser": {"attributes": {"name": "cons1", "pwd": "secret!"}},
"dataMode": "raw", "timestamp": 0, "version": 2, "time": 1398807575531},

{"collectionId": "ac62a200-9210-f53b-7114-a8f4cffb9a36", "id": "a2739d92-5f9d-f16c-8894-0f64b6f967a3",
"name": "consumer",
"description": "",
"url": "http://solar.local:8000/api/policymgr/mo/uni/tn-cons1.xml",
"method": "POST", "headers": "", "data":
"<fvTenant name=\"cons1\" status=\"modified\">\n
  <fvCtx name=\"cons1\"/>\n
  <!-- bridge domain -->\n
  <fvBD name=\"cons1\">\n
    <fvRsCtx tnFvCtxName=\"cons1\" />\n
    <fvSubnet ip=\"10.0.2.128/24\" scope='shared'>\n
  </fvBD>\n
  \n <!-- DNS Shared Service Contract Interface-->\n
  <vzCPIf name=\"consIf\">\n
    <vzRsIf tDn=\"uni/tn-prov1/brc-webCtrct\" >\n
  </vzRsIf>\n
</vzCPIf>\n \n
<fvAp name=\"cons1\">\n
  <fvAEPg name=\"APP\">\n
    <fvRsBd tnFvBDName=\"cons1\" />\n
    <fvRsNodeAtt tDn=\"topology/pod-1/node-101\" encap=\"vlan-4000\" instrImedcy=\"immediate\"
mode=\"regular\"/>\n
    <fvRsDomAtt tDn=\"uni/vmmp-VMware/dom-Datacenter\"/>\n
    <fvRsConsIf tnVzCPIfName=\"consIf\"/>\n
  </fvAEPg>\n
</fvAp>\n
</fvTenant>\n",
"dataMode": "raw", "timestamp": 0, "version": 2, "time": 1398818639692},

{"collectionId": "ac62a200-9210-f53b-7114-a8f4cffb9a36", "id": "c0bd866d-600a-4f45-46ec-6986398cbf78",
"name": "provider tenant only",
"description": "",
"url": "http://solar.local:8000/api/policymgr/mo/uni/tn-prov1.xml",
"method": "POST",
"headers": "",
"data":
"<fvTenant name=\"prov1\"><aaaDomainRef name=\"prov1\"/>

```

```

    \n
  </fvTenant>\n",
  "dataMode":"raw", "timestamp":0, "version":2, "time":1398818137518},

  {"collectionId":"ac62a200-9210-f53b-7114-a8f4cffb9a36", "id":"d433a213-e95d-646d-895e-3a9e2e2b7ba3",
  "name":"create RbacRule",
  "description":"",
  "url":"http://solar.local:8000/api/policymgr/mo/uni.xml",
  "method":"POST",
  "headers":"",
  "data":
  "<aaaRbacEp>\n
    <aaaRbacRule objectDn=\"uni/vmmp-VMware/dom-Datacenter\" domain=\"cons1\"/>\n
    <aaaRbacRule objectDn=\"uni/tn-prov1/brc-webCtrct\" domain=\"cons1\"/>\n
  </aaaRbacEp>\n",
  "dataMode":"raw", "timestamp":0, "version":2, "time":1414195420515},

  {"collectionId":"ac62a200-9210-f53b-7114-a8f4cffb9a36", "id":"d5c5d580-a11a-7c61-34ac-cbdac249157f",
  "name":"provider",
  "description":"",
  "url":"http://solar.local:8000/api/policymgr/mo/uni/tn-prov1.xml",
  "method":"POST",
  "headers":"",
  "data":
  "<fvTenant name=\"prov1\" status=\"modified\">\n
    <fvCtx name=\"prov1\"/>\n
    \n <!-- bridge domain -->\n
    <fvBD name=\"prov1\">\n
      <fvRsCtx tnFvCtxName=\"prov1\" />\n
    </fvBD>\n \n
    <vzFilter name='t0f0' >\n
      <vzEntry etherT='ip' dToPort='10' prot='6' name='t0f0e9' dFromPort='10'>
    </vzEntry>\n
    </vzFilter>\n \n
    <vzFilter name='t0f1'>\n
      <vzEntry etherT='ip' dToPort='209' prot='6' name='t0f1e8' dFromPort='109'>
    </vzEntry>\n
  </vzFilter>\n \n
    <vzBrCP name=\"webCtrct\" scope=\"global\">\n
      <vzSubj name=\"app\">\n
        <vzRsSubjFiltAtt tnVzFilterName=\"t0f0\"/>\n
        <vzRsSubjFiltAtt tnVzFilterName=\"t0f1\"/>\n
      </vzSubj>\n
    </vzBrCP>\n \n
    <fvAp name=\"prov1AP\">\n
      <fvAEPg name=\"Web\">\n
        <fvRsBd tnFvBDName=\"prov1\" />\n
        <fvRsNodeAtt tDn=\"topology/pod-1/node-17\" encap=\"vlan-4000\"
instrImedcy=\"immediate\" mode=\"regular\"/>\n
        <fvRsProv tnVzBrCPName=\"webCtrct\"/>\n
        <fvRsDomAtt tDn=\"uni/vmmp-VMware/dom-Datacenter\"/>\n
        <fvSubnet ip=\"10.0.1.128/24\" scope='shared' />\n
      </fvAEPg>\n
    </fvAp>\n
  </fvTenant>\n",
  "dataMode":"raw", "timestamp":0, "version":2, "time":1398818660457},

  {"collectionId":"ac62a200-9210-f53b-7114-a8f4cffb9a36", "id":"e8866493-2188-8893-8e0c-4ca0903b18b8",
  "name":"add user prov1",
  "description":"",
  "url":"http://solar.local:8000/api/policymgr/mo/uni/userext.xml",
  "method":"POST",
  "headers":"",
  "data":

```

```
"<aaaUserEp>\n
  <aaaUser name=\"prov1\" pwd=\"secret!\">
    <aaaUserDomain name=\"prov1\">
      <aaaUserRole name=\"tenant-admin\" privType=\"writePriv\"/>
      <aaaUserRole name=\"vmm-admin\" privType=\"writePriv\"/>
    </aaaUserDomain>
  </aaaUser>\n
  <aaaUser name=\"cons1\" pwd=\"secret!\">
    <aaaUserDomain name=\"cons1\">
      <aaaUserRole name=\"tenant-admin\" privType=\"writePriv\"/>
      <aaaUserRole name=\"vmm-admin\" privType=\"writePriv\"/>
    </aaaUserDomain>
  </aaaUser>\n
  <aaaDomain name=\"prov1\"/>\n
  <aaaDomain name=\"cons1\"/>\n
</aaaUserEp>\n",
"dataMode":"raw", "timestamp":0, "version":2, "time":1398820966635}}}
```

Enabling Port Security

About Port Security and ACI

The port security feature protects the ACI fabric from being flooded with unknown MAC addresses by limiting the number of MAC addresses learned per port. The port security feature support is available for physical ports, port channels, and virtual port channels.

Port Security Guidelines and Restrictions

The guidelines and restrictions are as follows:

- Port security is available per port.
- Port security is supported for physical ports, port channels, and virtual port channels (vPCs).
- Static and dynamic MAC addresses are supported.
- MAC address moves are supported from secured to unsecured ports and from unsecured ports to secured ports.
- The MAC address limit is enforced only on the MAC address and is not enforced on a MAC and IP address.
- Port security is not supported with the Fabric Extender (FEX).

Port Security and Learning Behavior

For non-vPC ports or port channels, whenever a learn event comes for a new endpoint, a verification is made to see if a new learn is allowed. If the corresponding interface has a port security policy not configured or disabled, the endpoint learning behavior is unchanged with what is supported. If the policy is enabled and the limit is reached, the current supported action is as follows:

- Learn the endpoint and install it in the hardware with a drop action.
- Silently discard the learn.

If the limit is not reached, the endpoint is learned and a verification is made to see if the limit is reached because of this new endpoint. If the limit is reached, and the learn disable action is configured, learning will be disabled in the hardware on that interface (on the physical interface or on a port channel or vPC). If the limit is reached and the learn disable action is not configured, the endpoint will be installed in hardware with a drop action. Such endpoints are aged normally like any other endpoints.

When the limit is reached for the first time, the operational state of the port security policy object is updated to reflect it. A static rule is defined to raise a fault so that the user is alerted. A syslog is also raised when the limit is reached.

In case of vPC, when the MAC limit is reached, the peer leaf switch is also notified so learning can be disabled on the peer. As the vPC peer can be rebooted any time or vPC legs can become unoperational or restart, this state will be reconciled with the peer so vPC peers do not go out of sync with this state. If they get out of sync, there can be a situation where learning is enabled on one leg and disabled on the other leg.

By default, once the limit is reached and learning is disabled, it will be automatically re-enabled after the default timeout value of 60 seconds.

Port Security at Port Level

In the APIC, the user can configure the port security on switch ports. Once the MAC limit has exceeded the maximum configured value on a port, all traffic from the exceeded MAC addresses is forwarded. The following attributes are supported:

- **Port Security Timeout**—The current supported range for the timeout value is from 60 to 3600 seconds.
- **Violation Action**—The violation action is available in protect mode. In the protect mode, MAC learning is disabled and MAC addresses are not added to the CAM table. Mac learning is re-enabled after the configured timeout value.
- **Maximum Endpoints**—The current supported range for the maximum endpoints configured value is from 0 to 12000. If the maximum endpoints value is 0, the port security policy is disabled on that port.

Protect Mode

The protect mode prevents further port security violations from occurring. Once the MAC limit exceeds the maximum configured value on a port, all traffic from excess MAC addresses will be dropped and further learning is disabled.

Configuring Port Security Using REST API

Configure the port security.

Example:

```
<polUni>
  <infraInfra>

    <l2PortSecurityPol name="testL2PortSecurityPol" maximum="10" violation="protect" timeout="300"/>

  <infraNodeP name="test">
    <infraLeafS name="test" type="range">
```

```

    <infraNodeBlk name="test" from_="101" to_="102"/>
  </infraLeafS>
  <infraRsAccPortP tDn="uni/infra/accportprof-test"/>
</infraNodeP>

  <infraAccPortP name="test">
    <infraHPortS name="pselc" type="range">
      <infraPortBlk name="blk"
        fromCard="1" toCard="1" fromPort="20" toPort="22">
        </infraPortBlk>
      <infraRsAccBaseGrp tDn="uni/infra/funcprof/accportgrp-testPortG" />
    </infraHPortS>
  </infraAccPortP>

  <infraFuncP>
    <infraAccPortGrp name="testPortG">
      <infraRsL2PortSecurityPol tnL2PortSecurityPolName="testL2PortSecurityPol"/>
      <infraRsAttEntP tDn="uni/infra/attentp-test" />
    </infraAccPortGrp>
  </infraFuncP>

  <infraAttEntityP name="test">
    <infraRsDomP tDn="uni/phys-mininet"/>
  </infraAttEntityP>
</infraInfra>
</polUni>

```

Enabling COOP Authentication

Overview

Council of Oracle Protocol (COOP) is used to communicate the mapping information (location and identity) to the spine proxy. A leaf switch forwards endpoint address information to the spine switch 'Oracle' using Zero Message Queue (ZMQ). COOP running on the spine nodes will ensure all spine nodes maintain a consistent copy of endpoint address and location information and additionally maintain the distributed hash table (DHT) repository of endpoint identity to location mapping database.

COOP data path communication provides high priority to transport using secured connections. COOP is enhanced to leverage the MD5 option to protect COOP messages from malicious traffic injection. The APIC controller and switches support COOP protocol authentication.

COOP protocol is enhanced to support two ZMQ authentication modes: strict and compatible.

- Strict mode: COOP allows MD5 authenticated ZMQ connections only.
- Compatible mode: COOP accepts both MD5 authenticated and non-authenticated ZMQ connections for message transportation.

Using COOP with Cisco APIC

To support COOP Zero Message Queue (ZMQ) authentication support across the Cisco Application Centric Infrastructure (ACI) fabric, the Application Policy Infrastructure Controller (APIC) supports the MD5 password and also supports the COOP secure mode.

COOP ZMQ Authentication Type Configuration—A new managed object, `coop:AuthP`, is added to the Data Management Engine (DME)/COOP database (`coop/inst/auth`). The default value for the attribute type is "compatible", and users have the option to configure the type to be "strict".

COOP ZMQ Authentication MD5 password—The APIC provides a managed object (`fabric:SecurityToken`), that includes an attribute to be used for the MD5 password. An attribute in this managed object, called "token", is a string that changes every hour. COOP obtains the notification from the DME to update the password for ZMQ authentication. The attribute token value is not displayed.

Guidelines and Limitations

Follow these guidelines and limitations:

- During an ACI fabric upgrade, the COOP strict mode is disallowed until all switches are upgraded. This protection prevents the unexpected rejection of a COOP connection that could be triggered by prematurely enabling the strict mode.

Configuring COOP Authentication Using the REST API

Configure a COOP authentication policy.

In the example, the strict mode is chosen.

Example:

```
https://172.23.53.xx/api/node/mo/uni/fabric/pol-default.xml
```

```
<coopPol type="strict">  
</coopPol>
```

Enabling Control Plane Policing

About Control Plane Policing

Control plane policing (CoPP) protects the control plane, which ensures network stability, reachability, and packet delivery.

This feature allows specification of parameters, for each protocol that can reach the control processor to be rate-limited using a policer. The policing is applied to all traffic destined to any of the IP addresses of the router or Layer 3 switch. A common attack vector for network devices is the denial-of-service (DoS) attack, where excessive traffic is directed at the device interfaces.

The Cisco Application Centric Infrastructure (ACI) leaf and spine switch NX-OS provides CoPP to prevent DoS attacks from impacting performance. Such attacks, which can be perpetrated either inadvertently or maliciously, typically involve high rates of traffic destined to the supervisor module of a Cisco ACI leaf and spine switch CPU or CPU itself.

The supervisor module of Cisco ACI leaf and spine switch switches divides the traffic that it manages into two functional components or planes:

- **Data plane**—Handles all the data traffic. The basic functionality of a Cisco NX-OS device is to forward packets from one interface to another. The packets that are not meant for the switch itself are called the transit packets. These packets are handled by the data plane.
- **Control plane**—Handles all routing protocol control traffic. These protocols, such as the Border Gateway Protocol (BGP) and the Open Shortest Path First (OSPF) Protocol, send control packets between devices. These packets are destined to router addresses and are called control plane packets.

The Cisco ACI leaf and spine switch supervisor module has a control plane and is critical to the operation of the network. Any disruption or attacks to the supervisor module will result in serious network outages. For example, excessive traffic to the supervisor module could overload and slow down the performance of the entire Cisco ACI fabric. Another example is a DoS attack on the Cisco ACI leaf and spine switch supervisor module that could generate IP traffic streams to the control plane at a very high rate, forcing the control plane to spend a large amount of time in handling these packets and preventing the control plane from processing genuine traffic.

Examples of DoS attacks are as follows:

- Internet Control Message Protocol (ICMP) echo requests
- IP fragments
- TCP SYN flooding

These attacks can impact the device performance and have the following negative effects:

- Reduced service quality (such as poor voice, video, or critical applications traffic)
- High route processor or switch processor CPU utilization
- Route flaps due to loss of routing protocol updates or keepalives
- Processor resource exhaustion, such as the memory and buffers
- Indiscriminate drops of incoming packets



Note Cisco ACI leaf and spine switches are by default protected by CoPP with default settings. This feature allows for tuning the parameters on a group of nodes based on customer needs.

Control Plane Protection

To protect the control plane, the Cisco NX-OS running on Cisco ACI leaf and spine switches segregates different packets destined for the control plane into different classes. Once these classes are identified, the Cisco NX-OS device polices the packets, which ensures that the supervisor module is not overwhelmed.

Control Plane Packet Types:

Different types of packets can reach the control plane:

- **Receive Packets**—Packets that have the destination address of a router. The destination address can be a Layer 2 address (such as a router MAC address) or a Layer 3 address (such as the IP address of a router interface). These packets include router updates and keepalive messages. Multicast packets can also be in this category where packets are sent to multicast addresses that are used by a router.
- **Exception Packets**—Packets that need special handling by the supervisor module. For example, if a destination address is not present in the Forwarding Information Base (FIB) and results in a miss, the supervisor module sends an ICMP unreachable packet back to the sender. IP packet with IP options are dropped by the supervisor.
- **Redirect Packets**—Packets that are redirected to the supervisor module. Features such as Dynamic Host Configuration Protocol (DHCP) snooping or dynamic Address Resolution Protocol (ARP) inspection redirect some packets to the supervisor module.
- **Glean Packets**—If a Layer 2 MAC address for a destination IP address is not present in the FIB, the supervisor module receives the packet and sends an ARP request to the host.

All of these different packets could be maliciously used to attack the control plane and overwhelm the Cisco ACI fabric. CoPP classifies these packets to different classes and provides a mechanism to individually control the rate at which the Cisco ACI leaf and spine switch supervisor module receives these packets.

Classification for CoPP:

For effective protection, the Cisco ACI leaf and spine switch NX-OS classifies the packets that reach the supervisor modules to allow you to apply different rate controlling policies based on the type of the packet. For example, you might want to be less strict with a protocol packet such as Hello messages but more strict with a packet that is sent to the supervisor module because the IP option is set.

Available Protocols:

Protocol	Description
Glean	With this protocol, when the bridge domain is in proxy mode, unknown unicast traffic received by the leaf switch is sent to the hardware proxy (the spine switch). The spine switch changes the eth-type of the packet to a special eth-type (0xfff2). When these packets reach the leaf switches through the fabric ports, the packets are classified under glean. The packets are sent to the leaf switch's CPU, and the leaf switch's CPU generates an ARP request for the connected external devices.
ToR Glean	ToR glean activates when an endpoint moves or is cleared because of link flap and does not update the source leaf switch's remote IP address endpoint entry. A packet egresses the source leaf switch with the destination leaf switch's TEP address. On the destination leaf switch, because of the missing local IP address entry, the packet gets sent to the leaf switch CPU to generate an ARP request for those IP addresses. These packets are classified under ToR glean.

Rate Controlling Mechanisms:

Once the packets are classified, the Cisco ACI leaf and spine switch NX-OS has different mechanisms to control the rate at which packets arrive at the supervisor module.

You can configure the following parameters for policing:

- **Committed information rate (CIR)**—Desired bandwidth, specified as a bit rate or a percentage of the link rate.
- **Committed burst (BC)**—Size of a traffic burst that can exceed the CIR within a given unit of time and not impact scheduling.

Default Policing Policies:

When the Cisco ACI leaf and spine switch is bootup, the platform setup pre-defined CoPP parameters for different protocols are based on the tests done by Cisco.

Guidelines and Limitations for CoPP

CoPP has the following configuration guidelines and limitations:

- We recommend that you use the default CoPP policy initially and then later modify the CoPP policies based on the data center and application requirements.
- Customizing CoPP is an ongoing process. CoPP must be configured according to the protocols and features used in your specific environment as well as the supervisor features that are required by the server environment. As these protocols and features change, CoPP must be modified.
- We recommend that you continuously monitor CoPP. If drops occur, determine if CoPP dropped traffic unintentionally or in response to a malfunction or attack. In either event, analyze the situation and evaluate the need to modify the CoPP policies.
- You must ensure that the CoPP policy does not filter critical traffic such as routing protocols or interactive access to the device. Filtering this traffic could prevent remote access to the Cisco ACI Leaf/Spine and require a console connection.
- Do not mis-configure CoPP pre-filter entries. CoPP pre-filter entries might impact connectivity to multi-pod configurations, remote leaf switches, and Cisco ACI Multi-Site deployments.
- You can use the APIC UI to be able to tune the CoPP parameters.
- Per interface per protocol is only supported on Leaf switches.
- FEX ports are not supported on per interface per protocol.
- For per interface per protocol the supported protocols are; ARP, ICMP, CDP, LLDP, LACP, BGP, STP, BFD, and OSPF.
- The TCAM entry maximum for per interface per protocol is 256. Once the threshold is exceeded a fault will be raised.

Configuring CoPP Using the REST API

Step 1 Configure a CoPP leaf profile:

Example:

```
<!-- api/node/mo/uni/.xml -->
<infraInfra>
  <coppleafProfile type="custom" name="mycustom">                                <!-- define copp leaf profile -->

    <coppleafGen1CustomValues bgpBurst="150" bgpRate="300"/>
  </coppleafProfile>
  <infraNodeP name="leafCopp">
    <infraLeafS name="leafs" type="range">
      <infraNodeBlk name="leaf1" from_"101" to_"101"/>
      <infraNodeBlk name="leaf3" from_"103" to_"103"/>
      <infraRsAccNodePGrp tDn="uni/infra/funcprof/accnodepgrp-myLeafCopp"/>
    </infraLeafS>
  </infraNodeP>
  <infraFuncP>
    <infraAccNodePGrp name="myLeafCopp">
      <infraRsLeafCoppProfile tnCoppLeafProfileName="mycustom"/>    <!-- bind copp leaf policy to leaf
                                                                    profile -->
    </infraAccNodePGrp>
  </infraFuncP>
</infraInfra>
```

Step 2 Configure a CoPP spine profile:

Example:

```
<!-- api/node/mo/uni/.xml -->
<infraInfra>
  <coppspineProfile type="custom" name="mycustomSpine">                        <!-- define copp leaf profile
-->
    <coppspineGen1CustomValues bgpBurst="150" bgpRate="300"/>
  </coppspineProfile>
  <infraSpineP name="spineCopp">
    <infraSpineS name="spines" type="range">
      <infraNodeBlk name="spine1" from_"104" to_"104"/>
      <infraRsSpineAccNodePGrp tDn="uni/infra/funcprof/spaccnodepgrp-mySpineCopp"/>
    </infraSpineS>
  </infraSpineP>
  <infraFuncP>
    <infraSpineAccNodePGrp name="mySpineCopp">
      <infraRsSpineCoppProfile tnCoppSpineProfileName="mycustomSpine"/> <!-- bind copp spine policy
to
                                                                    spine profile -->
    </infraSpineAccNodePGrp>
  </infraFuncP>
</infraInfra>
```

Configuring CoPP Per Interface Per Protocol Using REST API

Configure a CoPP per interface per protocol:

Example:

```

<polUni>
  <infraInfra>
    <infraNodeP name="default">
      <infraLeafS name="default" type="range">
        <infraNodeBlk name="default" to_"101" from_"101"/>
      </infraLeafS>
      <infraRsAccPortP tDn="uni/infra/accportprof-default"/>
    </infraNodeP>
    <infraAccPortP name="default">
      <infraHPortS name="regularPorts" type="range">
        <infraPortBlk name="blk1" toPort="7" fromPort="1" toCard="1" fromCard="1"/>
        <infraRsAccBaseGrp tDn="uni/infra/funcprof/accportgrp-copp"/>
      </infraHPortS>
    </infraAccPortP>

    <infraFuncP>
      <infraAccPortGrp name="copp">
        <infraRsCoppIfPol tnCoppIfPolName="pc"/>
      </infraAccPortGrp>
    </infraFuncP>

    <coppIfPol name = "pc" >
      <coppProtoClassP name = "test" matchProto="lldp,arp" rate="505" burst = "201"/>
      <coppProtoClassP name = "test1" matchProto="bgp" rate="500" burst = "200" />
    </coppIfPol>
  </infraInfra>
</polUni>

```

Configuring First Hop Security

About First Hop Security

First-Hop Security (FHS) features enable a better IPv4 and IPv6 link security and management over the layer 2 links. In a service provider environment, these features closely control address assignment and derived operations, such as Duplicate Address Detection (DAD) and Address Resolution (AR).

The following supported FHS features secure the protocols and help build a secure endpoint database on the fabric leaf switches, that are used to mitigate security threats such as MIM attacks and IP thefts:

- ARP Inspection—allows a network administrator to intercept, log, and discard ARP packets with invalid MAC address to IP address bindings.
- ND Inspection—learns and secures bindings for stateless autoconfiguration addresses in Layer 2 neighbor tables.
- DHCP Inspection—validates DHCP messages received from untrusted sources and filters out invalid messages.
- RA Guard—allows the network administrator to block or reject unwanted or rogue router advertisement (RA) guard messages.
- IPv4 and IPv6 Source Guard—blocks any data traffic from an unknown source.

- **Trust Control**—a trusted source is a device that is under your administrative control. These devices include the switches, routers, and servers in the Fabric. Any device beyond the firewall or outside the network is an untrusted source. Generally, host ports are treated as untrusted sources.

FHS features provide the following security measures:

- **Role Enforcement**—Prevents untrusted hosts from sending messages that are out the scope of their role.
- **Binding Enforcement**—Prevents address theft.
- **DoS Attack Mitigations**—Prevents malicious end-points to grow the end-point database to the point where the database could stop providing operation services.
- **Proxy Services**—Provides some proxy-services to increase the efficiency of address resolution.

FHS features are enabled on a per tenant bridge domain (BD) basis. As the bridge domain, may be deployed on a single or across multiple leaf switches, the FHS threat control and mitigation mechanisms cater to a single switch and multiple switch scenarios.

ACI FHS Deployment

Most FHS features are configured in a two-step fashion: firstly you define a policy which describes the behavior of the feature, secondly you apply this policy to a "domain" (being the Tenant Bridge Domain or the Tenant Endpoint Group). Different policies that define different behaviors can be applied to different intersecting domains. The decision to use a specific policy is taken by the most specific domain to which the policy is applied.

The policy options can be defined from the Cisco APIC GUI found under the Tenant_<name>Networking>Protocol Policies>First Hop Security tab.

Guidelines and Limitations

Follow these guidelines and limitations:

- Starting with release 3.1(1), FHS is supported with virtual Endpoints (AVS only).
- FHS is supported with both VLAN and VXLAN encapsulation.
- Any secured endpoint entry in the FHS Binding Table Database in **DOWN** state will get cleared after **18 Hours** of timeout. The entry moves to **DOWN** state when the front panel port where the entry is learned is link down. During this window of **18 Hours**, if the endpoint is moved to a different location and is seen on a different port, the entry will be gracefully moved out of **DOWN** state to **REACHABLE/STALE** as long as the endpoint is reachable from the other port it is moved from.
- When IP Source Guard is enabled, the IPv6 traffic that is sourced using IPv6 Link Local address as IP source address is not subject to the IP Source Guard enforcement (i.e. Enforcement of Source Mac <=> Source IP Bindings secured by IP Inspect Feature). This traffic is permitted by default irrespective of binding check failures.
- FHS is not supported on L3Out interfaces.
- FHS is not supported N9K-M12PQ based TORs.

- FHS in ACI Multi-Site is a site local capability therefore it can only be enabled in a site from the APIC cluster. Also, FHS in ACI Multi-Site only works when the BD and EPG is site local and not stretched across sites. FHS security cannot be enabled for stretched BD or EPGs.
- FHS is not supported on a Layer 2 only bridge domain.
- Enabling FHS feature can disrupt traffic for 50 seconds because the EP in the BD are flushed and EP Learning in the BD is disabled for 50 seconds.

Configuring FHS in APIC Using REST API

Before you begin

- The tenant and bridge domain must be configured.

Configure the FHS and Trust Control policies.

Example:

```
<polUni>
  <fvTenant name="Coke">
    <fhsBDPol name="bdpol15" ipInspectAdminSt="enabled-ipv6" srcGuardAdminSt="enabled-both"
raGuardAdminSt="enabled" status="">
      <fhsRaGuardPol name="raguard5" managedConfigCheck="true" managedConfigFlag="true"
otherConfigCheck="true" otherConfigFlag="true" maxRouterPref="medium" minHopLimit="3" maxHopLimit="15"
status=""/>
    </fhsBDPol>
    <fvBD name="bd3">
      <fvRsBDToFhs tnFhsBDPolName="bdpol15" status=""/>
    </fvBD>
  </fvTenant>
</polUni>

<polUni>
<fvTenant name="Coke">
  <fhsTrustCtrlPol name="trustctrl15" hasDhcpv4Server="true" hasDhcpv6Server="true"
hasIpv6Router="true" trustRa="true" trustArp="true" trustNd="true" />
  <fvAp name="wwwCokecom3">
    <fvAEPg name="test966">
      <fvRsTrustCtrl tnFhsTrustCtrlPolName="trustctrl15" status=""/>
    </fvAEPg>
  </fvAp>
</fvTenant>
</polUni>
```

Configuring 802.1x

802.1X Overview

802.1X defines a client-server based access control and authentication protocol that restricts unauthorized clients from connecting to a LAN through publicly accessible ports. The authentication server authenticates each client connected to a Cisco NX-OS device port.

Until the client is authenticated, 802.1X access control allows only Extensible Authentication Protocol over LAN (EAPOL) traffic through the port to which the client is connected. After authentication is successful, normal traffic can pass through the port.

The RADIUS distributed client/server system allows you to secure networks against unauthorized access. In the Cisco ACI implementation, RADIUS clients run on the ToRs and send authentication and accounting requests to a central RADIUS server that contains all user authentication and network service access information.

Host Support

The 802.1X feature can restrict traffic on a port with the following modes:

- **Single-host Mode**—Allows traffic from only one endpoint device on the 802.1X port. Once the endpoint device is authenticated, the APIC puts the port in the authorized state. When the endpoint device leaves the port, the APIC put the port back into the unauthorized state. A security violation in 802.1X is defined as a detection of frames sourced from any MAC address other than the single MAC address authorized as a result of successful authentication. In this case, the interface on which this security association violation is detected (EAPOL frame from the other MAC address) will be disabled. Single host mode is applicable only for host-to-switch topology and when a single host is connected to the Layer 2 (Ethernet access port) or Layer 3 port (routed port) of the APIC.
- **Multi-host Mode**—Allows multiple hosts per port but only the first one gets authenticated. The port is moved to the authorized state after the successful authorization of the first host. Subsequent hosts are not required to be authorized to gain network access once the port is in the authorized state. If the port becomes unauthorized when reauthentication fails or an EAPOL logoff message is received, all attached hosts are denied access to the network. The capability of the interface to shut down upon security association violation is disabled in multiple host mode. This mode is applicable for both switch-to-switch and host-to-switch topologies
- **Multi-Auth Mode**—Allows multiple hosts and all hosts are authenticated separately.



Note Each host must have the same EPG/VLAN information.

- **Multi-Domain Mode**—For separate data and voice domain. For use with IP-Phones.

Authentication Modes

ACI 802.1X supports the following authentication modes:

- **EAP**—The authenticator then sends an EAP-request/identity frame to the supplicant to request its identity (typically, the authenticator sends an initial identity/request frame followed by one or more requests for authentication information). When the supplicant receives the frame, it responds with an EAP-response/identity frame.
- **MAB**—MAC Authentication Bypass (MAB) is supported as the fallback authentication mode. MAB enables port-based access control using the MAC address of the endpoint. A MAB-enabled port can be dynamically enabled or disabled based on the MAC address of the device that connects to it. Prior to MAB, the endpoint's identity is unknown and all traffic is blocked. The switch examines a single packet to learn and authenticate the source MAC address. After MAB succeeds, the endpoint's identity is known and all traffic from that endpoint is allowed. The switch performs source MAC address filtering to help ensure that only the MAB-authenticated endpoint is allowed to send traffic.

Guidelines and Limitations

802.1X port-based authentication has the following configuration guidelines and limitations:

- The Cisco ACI supports 802.1X authentication only on physical ports.
- The Cisco ACI does not support 802.1X authentication on port channels or subinterfaces.
- The Cisco ACI supports 802.1X authentication on member ports of a port channel but not on the port channel itself.
- Member ports with and without 802.1X configuration can coexist in a port channel. However, you must ensure the identical 802.1X configuration on all the member ports in order for channeling to operate with 802.1X
- When you enable 802.1X authentication, supplicants are authenticated before any other Layer 2 or Layer 3 features are enabled on an Ethernet interface.
- 802.1X is supported only on a leaf chassis that is EX or FX type.
- 802.1X is only supported Fabric Access Ports. 802.1X is not supported on Port-Channels, or Virtual-Port-Channels.
- IPv6 is not supported for dot1x clients in the 3.2(1) release.
- While downgrading to earlier releases especially in cases where certain interface config (host mode and auth type) is unsupported in that release, dot1x authentication type defaults to none. Host-mode would need to be manually re-configured to either single host/multi host depending on whatever is desired. This is to ensure that the user configures only the supported modes/auth-types in that release and doesn't run into unsupported scenarios.
- Multi-Auth supports 1 voice client and multiple data clients (all belonging to same data vlan/epg).
- Fail-epg/vlan under 802.1X node authentication policy is a mandatory configuration.
- Multi-domain more than 1 voice and 1 data client puts the port in security disabled state.
- The following platforms are not supported for 802.1X:
 - N9K-C9396PX
 - N9K-M12PQ
 - N9K-C93128TX

- N9K-M12PQ

Configuration Overview

The 802.1X and RADIUS processes are started only when enabled by APIC. Internally, this means dot1x process is started when 802.1X Inst MO is created and radius process is created when radius entity is created. Dot1x based authentication must be enabled on each interface for authenticating users connected on that interface otherwise the behavior is unchanged.

RADIUS server configuration is done separately from dot1x configuration. RADIUS configuration defines a list of RADIUS servers and a way to reach them. Dot1x configuration contains a reference to RADIUS group (or default group) to use for authentication.

Both 802.1X and RADIUS configuration must be done for successful authentication. Order of configuration is not important but if there is no RADIUS configuration then 802.1X authentication cannot be successful.

Configuring 802.1X Node Authentication Using the REST API

Configure a 802.1X node authentication policy:

Example:

```
<polUni>
<infraInfra>
  <l2NodeAuthPol annotation="" descr="" dn="uni/infra/nodeauthpol-802-node-2"
failAuthEpg="tn-t2,ap-ap,epg-epg1" failAuthVlan="vlan-2078" name="802-node-2" nameAlias="" ownerKey=""
ownerTag="">
  <l2RsAaaRadiusProviderGroup annotation="" tDn="uni/userext/radiusext/radiusprovidergroup-radius-grp"/>
</l2NodeAuthPol>
</infraInfra>
</polUni>
```

Modify:

```
<polUni>
<infraInfra>
  <l2NodeAuthPol annotation="" descr="" dn="uni/infra/nodeauthpol-802-node-2"
failAuthEpg="tn-t2,ap-ap,epg-epg1" failAuthVlan="vlan-2066" name="802-node-2" nameAlias="" ownerKey=""
ownerTag="" status="deleted">
  <l2RsAaaRadiusProviderGroup annotation="" tDn="uni/userext/radiusext/radiusprovidergroup-radius-grp"/>
</l2NodeAuthPol>
</infraInfra>
</polUni>
```

Delete:

```
<polUni>
<infraInfra>
  <l2NodeAuthPol annotation="" descr="" dn="uni/infra/nodeauthpol-802-node-2"
failAuthEpg="tn-t2,ap-ap,epg-epg1" failAuthVlan="vlan-2078" name="802-node-2" nameAlias="" ownerKey=""
ownerTag="" status="deleted">
  <l2RsAaaRadiusProviderGroup annotation="" tDn="uni/userext/radiusext/radiusprovidergroup-radius-grp"
status="deleted"/>
</l2NodeAuthPol>
</infraInfra>
</polUni>
```

Configuring 802.1X Port Authentication Using the REST API

Create a 802.1X port authentication policy:

Example:

```
<polUni>
<infraInfra>
  <l2PortAuthPol adminSt="enabled" annotation="" descr="" dn="uni/infra/portauthpol-test21"
hostMode="multi-auth" name="test21" nameAlias="" ownerKey="" ownerTag="">
  <l2PortAuthCfgPol annotation="" macAuth="bypass" maxReauthReq="2" maxReq="2" reAuthPeriod="3600"
serverTimeout="30" suppTimeout="30" txPeriod="30"/>
  </l2PortAuthPol>
</infraInfra>
</polUni>
```

Modify:

```
<polUni>
<infraInfra>
  <l2PortAuthPol adminSt="enabled" annotation="" descr="" dn="uni/infra/portauthpol-test21"
hostMode="multi-domain" name="test21" nameAlias="" ownerKey="" ownerTag="" >
  <l2PortAuthCfgPol annotation="" macAuth="eap" maxReauthReq="2" maxReq="2" reAuthPeriod="3600"
serverTimeout="30" suppTimeout="30" txPeriod="30"/>
  </l2PortAuthPol>
</infraInfra>
</polUni>
```

Delete:

```
<polUni>
<infraInfra>
  <l2PortAuthPol adminSt="enabled" annotation="" descr="" dn="uni/infra/portauthpol-test21"
hostMode="multi-host" name="test21" nameAlias="" ownerKey="" ownerTag="" status="deleted">
  <l2PortAuthCfgPol annotation="" macAuth="bypass" maxReauthReq="2" maxReq="2" reAuthPeriod="3600"
serverTimeout="30" suppTimeout="30" txPeriod="30" status="deleted"/>>
  </l2PortAuthPol>
</infraInfra>
</polUni>
```



CHAPTER 19

Creating Quota Management

- [About APIC Quota Management Configuration, on page 491](#)
- [Creating a Quota Management Configuration Using the REST API, on page 491](#)

About APIC Quota Management Configuration

Starting in the Cisco Application Policy Infrastructure Controller (APIC) Release 2.3(1), there are limits on number of objects a tenant admin can configure. This enables the admin to limit what managed objects that can be added under a given tenant or globally across tenants.

This feature is useful when you want to limit any tenant or group of tenants from exceeding ACI maximums per leaf or per fabric or unfairly consuming a majority of available resources, potentially affecting other tenants on the same fabric.

Creating a Quota Management Configuration Using the REST API

This procedure explains how to create a quota management configuration using the REST API.

SUMMARY STEPS

1. Create a quota management configuration using the REST API:

DETAILED STEPS

Create a quota management configuration using the REST API:

Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- /api/node/mo/.xml -->
<polUni>

  <quotaCont>

    <quotaConf class="fvBD" containerDn="uni/tn-green" maxNum="10" exceedAction="fault"/>
    <quotaConf class="fvBD" containerDn="uni/tn-baz" maxNum="100" exceedAction="fail"/>
  
```

```
</quotaCont>  
</polUni>
```



CHAPTER 20

Configuring a Forwarding Scale Profile Policy

- [Forwarding Scale Profile Policy Overview, on page 493](#)
- [Supported Platforms, on page 495](#)
- [Guidelines and Limitations, on page 496](#)
- [Configuring the Forwarding Scale Profile Policy Using the REST API, on page 497](#)

Forwarding Scale Profile Policy Overview

The Forwarding Scale Profile policy provides different scalability options. For example:

- **Dual Stack**—provides scalability of up to 12,000 endpoints for IPv6 configurations and up to 24,000 endpoints for IPv4 configurations.
- **High LPM**—provides scalability similar to the dual-stack profile, except that the longest prefix match (LPM) scale is 128,000 and the policy scale is 8,000.
- **IPv4 Scale**—enables systems with no IPv6 configurations to increase scalability to 48,000 IPv4 endpoints.
- **High Dual Stack**—provides scalability of up to 64,000 MAC endpoints and 64,000 IPv4 endpoints. IPv6 endpoint scale can be 24,000/48,000, depending on the switch hardware model.



Note With Cisco Application Policy Infrastructure Controller (APIC) release 3.2(1), depending on your leaf switch hardware, a Forwarding Scale Profile with the **High Dual Stack** option has different scales; for example:

- For Cisco Nexus 9000 Series leaf switches with FX in the switch name, the high dual-stack option has scalability of 48,000 IPv6 endpoints instead of 24,000 and 128,000 policies instead of 8,000.
- For Cisco Nexus 9000 Series leaf switches with EX in the switch name, the high dual-stack option has the same scale values as with earlier Cisco APIC releases.

See the following table for more details.

Table 13: Forwarding Scale Profile Policy Scalability

Forwarding Scale Profile Policy Options	Leaf Switches with EX and FX2 Names	Leaf Switches with FX Names
Dual Stack	<ul style="list-style-type: none"> • EP MAC: 24,000 • EP IPv4: 24,000 • EP IPv6: 12,000 • LPM: 20,000 • Policy: 64,000 • Multicast: 8,000 	Has the same scalability numbers as Dual Stack scale on earlier switches.
High Dual Stack	<ul style="list-style-type: none"> • EP MAC: 64,000 • EP IPv4: 64,000 • EP IPv6: 24,000 • LPM: 38,000 • Policy: 8,000 • Multicast: 512 	<ul style="list-style-type: none"> • EP MAC: 64,000 • EP IPv4: 64,000 • EP IPv6: 48,000 • LPM: 38,000 • Policy: 128,000 • Multicast: 32,000
High LPM	Provides scalability similar to the dual-stack profile, except that the longest prefix match (LPM) scale is 128,000 and the policy scale is 8,000.	Has the same scalability numbers as on earlier switches.
IPv4 Scale	<ul style="list-style-type: none"> • EP MAC: 48,000 • EP IPv4: 48,000 • EP IPv6: 0 • LPM: 38,000 • Policy: 64,000 • Multicast: 8,000 	Has the same scalability numbers as IPv4 scale on earlier switches.

**Note**

- For Cisco Nexus 9000 Series leaf switches with FX2 in the switch name, the scale values are same as those of EX series switches.
- Because the IPv4 forwarding scale profile policy does not support IPv6 configurations, all IPv6 configurations must be removed from switches configured with the IPv4 forwarding scale profile policy.
- For leaf switch models with EX at the end of the switch name, because the high dual stack profile has reduced-scale support for contract policies (8,000), the contracts scale must be reduced accordingly prior to deploying that profile.
- Before migrating to minimal tenant multicast scale leaf profiles, such as high dual stack, we recommend that you first disable Layer 2 IGMP snooping-, Layer 3 IGMP-, and PIM-related configurations to prevent having a stale multicast state in your hardware.
- Applying a scale profile to a node requires a manual reload of that node. Any unsupported switches are ignored. The following switches are supported:
 - Cisco Nexus 9300-EX series switches
 - Cisco Nexus 9300-FX series switches
 - Cisco Nexus 9300-FX2 series switches
- vPCs associated with different scale profile settings are not supported. The vPC members must be configured with the same scale profile settings.

Supported Platforms

This section provides forwarding scale profiles hardware support information for Release 4.1(1).

The following table summarizes platform support for each forwarding scale profile.

Table 14: Supported Switches

Scale Profile	Supported on EX and FX2 Switches	Supported on FX Switches
Dual Stack	Yes	Yes
High Dual Stack	Yes	Yes Different scale numbers than EX/FX2.
High LPM	Yes	Yes
IPv4 Scale	Yes	Yes

- Forwarding scale profiles are supported on the following switches:
 - Cisco Nexus 9300-EX series switches
 - Cisco Nexus 9300-FX series switches

- Cisco Nexus 9300-FX2 series switches
- Switches not listed here do not support forwarding scale profiles in this release.

Guidelines and Limitations

- When downgrading to a release that does not support one or more switches in your current fabric, keep the following in mind:
 - If you downgrade your fabric to a release where one or more of your current switches are not supported, those switches will become inactive in the fabric.
 - If you later upgrade the fabric to a release where the switch is supported again, the APIC will not regain complete details about the switch. In this case, you will need to explicitly remove the switch from the APIC and then re-add it to the fabric.
- When downgrading to a release that does not support one or more of your current forwarding scale profiles, the default forwarding scale profile will be configured on the switch. You must reduce the configurations on the switch to fit the default profile before the upgrade.
- Because the IPv4 Scale forwarding scale profile does not support IPv6 configurations, you must remove all IPv6 configurations from the switches that need to be configured with the IPv4 Scale profile.
- Before switching between forwarding scale profiles, the configurations on the switch must be reduced appropriately and thoroughly verified so that scale parameters of the target profile are not exceeded.

For example, for switch models with EX at the end of the switch name, because the High Dual Stack profile has reduced scale support for contract policies, you must reduce the contracts scale accordingly before deploying that profile.
- Before migrating to minimal tenant multicast scale leaf profiles, such as High Dual Stack, we recommend that you first disable Layer 2 IGMP snooping, Layer 3 IGMP, and PIM-related configurations to prevent having a stale multicast state in your hardware.
- Applying a forwarding scale profile to a node requires a manual reload of that node. Any unsupported switches are ignored.
- vPCs associated with different forwarding scale profile settings are not supported. You must configure the vPC members with the same profile settings.
- With the default deny model in Cisco ACI, the configured tenant or VRFs have implicit rules that consume several TCAM entries for each VRF. With an increase in the number of VRFs configured on a single switch, these TCAM entries that are used per VRF also count toward the overall policy TCAM usage.
- Beginning with Release 4.1(1), the policy count updates that are reported by the leaf switches to the Cisco APIC through the MO `actrlRuleHit5min` is set to 0 for the High Dual Stack profile for all platforms.
- Beginning with Release 4.2(1), the policy count updates that are reported by the leaf switches to the Cisco APIC through the MO `actrlRuleHit5min` is set to 0 for the High Policy profile on the Cisco Nexus 93180YC-FX switch.
- Beginning with Release 4.2(2), the policy count updates that are reported by the leaf switches to the Cisco APIC through the MO `actrlRuleHit5min` is set to 0 for the High Policy profile on the Cisco Nexus 93600CD-GX switch.

- Beginning with Release 4.2(31), the policy count updates that are reported by the leaf switches to the Cisco APIC through the MO `actrlRuleHit5min` is set to 0 for the High Policy profile on the Cisco Nexus 9364C-GX switch.
- If you need to clear the configurations on the switch, we recommend using the `setup-clean-config.sh -k` command. The command will clear all configurations on the switch, except the forwarding scale profile and port profile configurations.

Configuring the Forwarding Scale Profile Policy Using the REST API

The forwarding scale profile policy requires supported switches. For a list of supported switches, see the supported platforms section. The switches that support the forwarding scale profile policy must be manually reloaded after the forwarding scale profile policy is applied.

The Forwarding Scale Profile policy provides different scalability options. For more information on the scalability options, see [Forwarding Scale Profile Policy Overview](#), on page 493.

This section explains how to create a forwarding scale profile policy and apply it to a leaf profile using the REST API.

To apply a forwarding scale profile policy with IPv4 scaling, send a post with XML similar to the following example:

Example:

```
<polUni>
  <infraInfra>
    <topoctrlFwdScaleProfilePol name="sampleFwdScaleProf" profType="ipv4"/>
    <infraAccNodePGrp name="sampleNodePolGrp">
      <infraRsTopoctrlFwdScaleProfPol tnTopoctrlFwdScaleProfilePolName="sampleFwdScaleProf"/>
    </infraAccNodePGrp>
    <infraNodeP name="nodeProf_101">
      <infraLeafS name="leafS_101" type="range">
        <infraNodeBlk name="test" from_"101" to_"101"/>
        <infraRsAccNodePGrp tDn="uni/infra/funcprof/accnodepgrp-sampleNodePolGrp "/>
      </infraLeafS>
    </infraNodeP>
  </infraInfra>
</polUni>
```

To take effect, the switches that support the forwarding scale profile policy must be manually reloaded after the forwarding scale profile policy is applied.

